Introduction

Scaler is an online tech-versity offering intensive computer science & Data Science courses through live classes delivered by tech leaders and subject matter experts. The meticulously structured program enhances the skills of software professionals by offering a modern curriculum with exposure to the latest technologies. It is a product by InterviewBit.

Problem Statement

You are working as a data scientist with the analytics vertical of Scaler, focused on profiling the best companies and job positions to work for from the Scaler database. You are provided with the information for a segment of learners and tasked to cluster them on the basis of their job profile, company, and other features. Ideally, these clusters should have similar characteristics.

Approach to solve Problem

The goal is to create clusters with similar characteristics, helping to identify patterns and preferences among learners regarding job profiles and companies. Steps needed:

1. Data available in csv file. Load and Read the data.
2. Clean and preprocess the data to handle missing values, outliers, and inconsistencies.
3. Identify relevant features or create new features that can help in clustering learners.
4. Do Exploratory Data Analysis(EDA) to understand distribution of features, relationships between variables, and potential clusters in the data.
5. Visualize data wherever necessary. Make sure data is ready for modeling.
6. As stated, do manual clustering based on conditions of features
7. Choose appropriate Clustering Algorithm and apply it to pre-processed data to create clsuters.
8. Evaluate quality of clusters using metrics, visualize the clusters and their characteristics using plots.
9. Summarize the findings, insights, and recommendations based on the clustered learner profiles.

In [1]:
```
# Import libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
import warnings
warnings.filterwarnings("ignore")
```

In [3]:
```
# Load the dataset
data = pd.read_csv("https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000/
```

In [4]:
```python
# Snippet of dataset
data.head(2)
```

Out[4]:

	Unnamed: 0	company_hash	email_hash	orgyear	ct
0	0	atrgxnnt xzaxv	6de0a4417d18ab14334c3f43397fc13b30c35149d70c05...	2016.0	110000
1	1	qtrxvzwt xzegwgbb rxbxnta	b0aaf1ac138b53cb6e039ba2c3d6604a250d02d5145c10...	2018.0	44999

In [5]:
```python
# shape of dataset
data.shape

# There are 205843 entries with 7 features
```

Out[5]:
```
(205843, 7)
```

In [6]:
```python
# Information about dataset
data.info()

# There are 7 features : "Unnamed:0", "company_hash", "email_hash", "orgyear", "ctc
# On observing Non-Null count column values, it can be inferred that there are miss
# features such as "Unnamed: 0" and "ctc" has "int64" Dtype
# features such as "company_hash", "email_hash" and "job_position" has "object" Dty
# features such as "orgyear" and "ctc_updated_year" has "float64" Dtype
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 205843 entries, 0 to 205842
Data columns (total 7 columns):
 #   Column            Non-Null Count   Dtype
---  ------            --------------   -----
 0   Unnamed: 0        205843 non-null  int64
 1   company_hash      205799 non-null  object
 2   email_hash        205843 non-null  object
 3   orgyear           205757 non-null  float64
 4   ctc               205843 non-null  int64
 5   job_position      153281 non-null  object
 6   ctc_updated_year  205843 non-null  float64
dtypes: float64(2), int64(2), object(3)
memory usage: 11.0+ MB
```

In [7]:
```python
# Creating a copy of dataframe
data_copy = data.copy(deep = True)
```

In [8]:
```python
# count of missing values in column
display(data[data.columns[data.isnull().any()]].isnull().sum())
# percentage of missing values
display(data[data.columns[data.isnull().any()]].isnull().sum()*100/data.shape[0])
```

```
company_hash       44
orgyear            86
job_position    52562
dtype: int64
company_hash     0.021376
orgyear          0.041779
job_position    25.534995
dtype: float64
```

Analysis of each feature

In [9]:
```
data["Unnamed: 0"].value_counts()

# As this feature just contains serial numbers. It is not relevant and not required
```

Out[9]:
```
0         1
137694    1
137684    1
137685    1
137686    1
         ..
68713     1
68714     1
68715     1
68716     1
206922    1
Name: Unnamed: 0, Length: 205843, dtype: int64
```

In [10]:
```
data.drop(columns = "Unnamed: 0",inplace = True)
```

In [11]:
```
data["ctc_updated_year"].value_counts()

# ctc_updated_year showing years 2015 to 2020.
# Datatype of this feature should be in datetime format for analysis.
# But for modeling, let it be as numerical.
```

Out[11]:
```
2019.0    68688
2021.0    64976
2020.0    49444
2017.0     7561
2018.0     6746
2016.0     5501
2015.0     2927
Name: ctc_updated_year, dtype: int64
```

In [12]:
```
data["orgyear"].value_counts()

# orgyear is the employement start date
# Datatype of this feature should be in datetime format for analysis.
# But for modeling, let it be as numerical.
```

Out[12]:
```
2018.0    25256
2019.0    23427
2017.0    23239
2016.0    23043
2015.0    20610
          ...
2107.0        1
1972.0        1
2101.0        1
208.0         1
200.0         1
Name: orgyear, Length: 77, dtype: int64
```

In [13]:
```
data["email_hash"].value_counts()

# It can be observe that few emails have been occur frequently. It need to be analy
```

Out[13]:
```
bbace3cc586400bbc65765bc6a16b77d8913836cfc98b77c05488f02f5714a4b    10
6842660273f70e9aa239026ba33bfe82275d6ab0d20124021b952b5bc3d07e6c     9
298528ce3160cc761e4dc37a07337ee2e0589df251d73645aae209b010210eee     9
3e5e49daa5527a6d5a33599b238bf9bf31e85b9efa9a94f1c88c5e15a6f31378     9
b4d5afa09bec8689017d8b29701b80d664ca37b83cb883376b2e95191320da66     8
                                                                    ..
bb2fe5e655ada7f7b7ac4a614db0b9c560e796bdfcaa4e5367e69eedfea93876     1
d6cdef97e759dbf1b7522babccbbbd5f164a75d1b4139e02c945958720f1ed79     1
700d1190c17aaa3f2dd9070e47a4c042ecd9205333545dbfaee0f85644d00306     1
c2a1c9e4b9f4e1ed7d889ee4560102c1e2235b2c1a0e59cea95a6fe55c658407     1
0bcfc1d05f2e8dc4147743a1313aa70a119b41b30d4a1f7e738a6a87d3712c31     1
Name: email_hash, Length: 153443, dtype: int64
```

In [14]:
```python
# Analyse the first email to know the detail
data[data["email_hash"] == "bbace3cc586400bbc65765bc6a16b77d8913836cfc98b77c05488f0

# It has been observed that learner with this email is working in same company and
# in job positions but ctc updated only once.
```

Out[14]:

	company_hash	email_hash	orgyear	ctc	job_
24109	oxej ntwyzgrgsxto rxbxnta	bbace3cc586400bbc65765bc6a16b77d8913836cfc98b7...	2018.0	720000	
45984	oxej ntwyzgrgsxto rxbxnta	bbace3cc586400bbc65765bc6a16b77d8913836cfc98b7...	2018.0	720000	
72315	oxej ntwyzgrgsxto rxbxnta	bbace3cc586400bbc65765bc6a16b77d8913836cfc98b7...	2018.0	720000	
102915	oxej ntwyzgrgsxto rxbxnta	bbace3cc586400bbc65765bc6a16b77d8913836cfc98b7...	2018.0	720000	
117764	oxej ntwyzgrgsxto rxbxnta	bbace3cc586400bbc65765bc6a16b77d8913836cfc98b7...	2018.0	720000	Data
121483	oxej ntwyzgrgsxto rxbxnta	bbace3cc586400bbc65765bc6a16b77d8913836cfc98b7...	2018.0	660000	
124476	oxej ntwyzgrgsxto rxbxnta	bbace3cc586400bbc65765bc6a16b77d8913836cfc98b7...	2018.0	660000	
144479	oxej ntwyzgrgsxto rxbxnta	bbace3cc586400bbc65765bc6a16b77d8913836cfc98b7...	2018.0	660000	
152801	oxej ntwyzgrgsxto rxbxnta	bbace3cc586400bbc65765bc6a16b77d8913836cfc98b7...	2018.0	660000	
159835	oxej ntwyzgrgsxto rxbxnta	bbace3cc586400bbc65765bc6a16b77d8913836cfc98b7...	2018.0	660000	

In [15]:
```python
# Analyse the first email to know the detail
data[data["email_hash"] == "6842660273f70e9aa239026ba33bfe82275d6ab0d20124021b952b5
```

```
# It has been observed that learner with this email is working in same company with
# He only gained multiple job positions.
```

Out[15]:

	company_hash	email_hash	orgyear	ctc	job
9857	ihvrwgbb	6842660273f70e9aa239026ba33bfe82275d6ab0d20124...	2017.0	2400000	QA
10002	ihvrwgbb	6842660273f70e9aa239026ba33bfe82275d6ab0d20124...	2017.0	2400000	
10583	ihvrwgbb	6842660273f70e9aa239026ba33bfe82275d6ab0d20124...	2017.0	2400000	
12784	ihvrwgbb	6842660273f70e9aa239026ba33bfe82275d6ab0d20124...	2017.0	2400000	
20715	ihvrwgbb	6842660273f70e9aa239026ba33bfe82275d6ab0d20124...	2017.0	2400000	
138253	ihvrwgbb	6842660273f70e9aa239026ba33bfe82275d6ab0d20124...	2017.0	2000000	
159251	ihvrwgbb	6842660273f70e9aa239026ba33bfe82275d6ab0d20124...	2017.0	2000000	
165343	ihvrwgbb	6842660273f70e9aa239026ba33bfe82275d6ab0d20124...	2017.0	2000000	
178749	ihvrwgbb	6842660273f70e9aa239026ba33bfe82275d6ab0d20124...	2017.0	2000000	

Now, as our goal is to focus on profiling best companies and job positions for learners. It majorly depends on company profile, job_position and ctc. Feature "email_hash" is not contributing much in determining our goal and hence, not much relevant. Therefore, deleting this feature would not affect much in modeling

In [16]:

```
data.drop(columns = "email_hash",inplace = True)
```

In [17]:

```
data.info()

# company_hash represents Current employer of the learner
# CTC- Current CTC
# Job_position- Job profile in the company
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 205843 entries, 0 to 205842
Data columns (total 5 columns):
 #   Column            Non-Null Count   Dtype  
---  ------            --------------   -----  
 0   company_hash      205799 non-null  object 
 1   orgyear           205757 non-null  float64
 2   ctc               205843 non-null  int64  
 3   job_position      153281 non-null  object 
 4   ctc_updated_year  205843 non-null  float64
dtypes: float64(2), int64(1), object(2)
memory usage: 7.9+ MB
```

Check for duplicacy

In [18]:

```
data.duplicated().sum()

# 17470 entries are duplicates.
```

Out[18]:

```
17470
```

In [19]:
```
data.duplicated().sum()/len(data)*100
```
Out[19]:
```
8.48705081056922
```
In [20]:
```
# Delete duplicates
data.drop_duplicates(inplace=True)
```
In [21]:
```
# Check for duplicacy again
data.duplicated().sum()
```
Out[21]:
```
0
```
Checking for missing values and Prepare data for KNN/ Mean Imputation.

In [22]:
```
data.isnull().sum()
```
Out[22]:
```
company_hash            44
orgyear                 86
ctc                      0
job_position         43513
ctc_updated_year         0
dtype: int64
```
In [23]:
```
((data.isnull().sum()/len(data))*100).sort_values(ascending = False)

#Three features have missing values.
# "job_position" has 23% missing data, "orgyear" has 0.04% missing data and "compar
# As "job_position" feature is relevant in modeling and it has 25% missing data. It
# technique to fill missing data.
```
Out[23]:
```
job_position        23.099383
orgyear              0.045654
company_hash         0.023358
ctc                  0.000000
ctc_updated_year     0.000000
dtype: float64
```
In [24]:
```
data["job_position"].value_counts()

# As job position holds maximum missing value and this feature contains text
# therefore, need to clean text data first, then compute missing data again
```
Out[24]:
```
Backend Engineer                 40324
FullStack Engineer               22900
Other                            16184
Frontend Engineer                10110
Engineering Leadership            6831
                                 ...  
ayS                                  1
Principal Product Engineer           1
Senior Director of Engineering       1
Seller Support Associate             1
Android Application developer        1
Name: job_position, Length: 1017, dtype: int64
```
Cleaning text data

In [25]:
```
import re
```

In [26]:
```
def remove_special_char(string):
    new_string=re.sub('[^A-Za-z0-9 ]+', '', string)
    return new_string
```

In [27]:
```
data["company_hash"] = data["company_hash"].apply(lambda x: remove_special_char(str
```

In [28]:
```
data["job_position"] = data["job_position"].apply(lambda x: remove_special_char(str
```

In [29]:
```
# Check again missing values in dataset
((data.isnull().sum()/len(data))*100).sort_values(ascending = False)

# Only orgyear contains missing values, that to only 0.04%
# we can do Knn imputation, but as value is too small,we can delete those rows also
```

Out[29]:
```
orgyear             0.045654
company_hash        0.000000
ctc                 0.000000
job_position        0.000000
ctc_updated_year    0.000000
dtype: float64
```

In [30]:
```
data.dropna(inplace = True)
```

In [31]:
```
data.isnull().sum().sum()
```

Out[31]:
```
0
```

Statistical Analysis

In [32]:
```
# statistical Analysis of all numerical feature
data.describe().transpose()
```

Out[32]:

	count	mean	std	min	25%	50%	75%	
orgyear	188287.0	2.014615e+03	6.644463e+01	0.0	2013.0	2016.0	2018.0	2.
ctc	188287.0	2.387833e+06	1.221085e+07	2.0	600000.0	1000000.0	1750000.0	1.
ctc_updated_year	188287.0	2.019576e+03	1.343303e+00	2015.0	2019.0	2020.0	2021.0	2.

In [33]:
```
# statistical Analysis of all categorical features
data.describe(include = "object").transpose()
```

Out[33]:

	count	unique	top	freq
company_hash	188287	37275	nvnv wgzohrnvzwj otqcxwto	4282
job_position	188287	1006	nan	43489

Univariate Analysis

In [34]:
```
from IPython.display import display, HTML
CSS = """
.output {
 flex-direction: row;
}
```

```
"""
HTML('<style>{}</style>'.format(CSS))
```

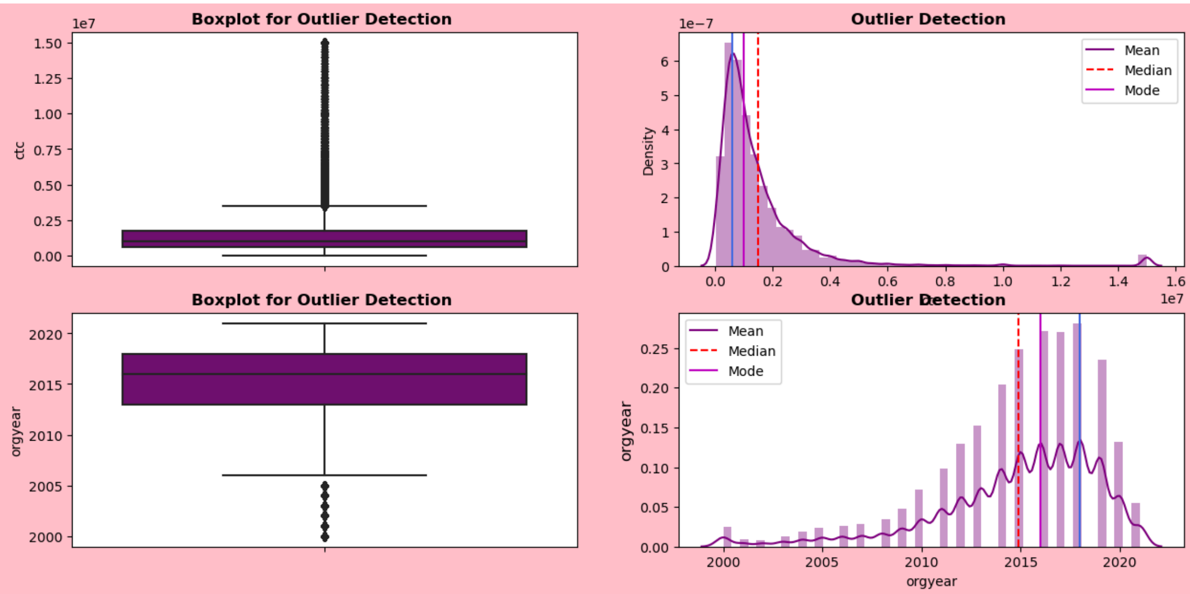
Out[34]:

In [35]:

```
# The values of features "ctc" and "orgyear" have very high range and sparse. To vi
# clip the data from 1 percentile to 99 percentile
data_ctc_orgyear = pd.DataFrame()
data_ctc_orgyear["ctc"] = data['ctc'].clip(data["ctc"].quantile(0.01),data["ctc"].c
data_ctc_orgyear["ctc"] = data_ctc_orgyear['ctc'].astype('int')
data_ctc_orgyear["orgyear"] = data['orgyear'].clip(data["orgyear"].quantile(0.01),d
data_ctc_orgyear["orgyear"] = data_ctc_orgyear['orgyear'].astype('int')
```

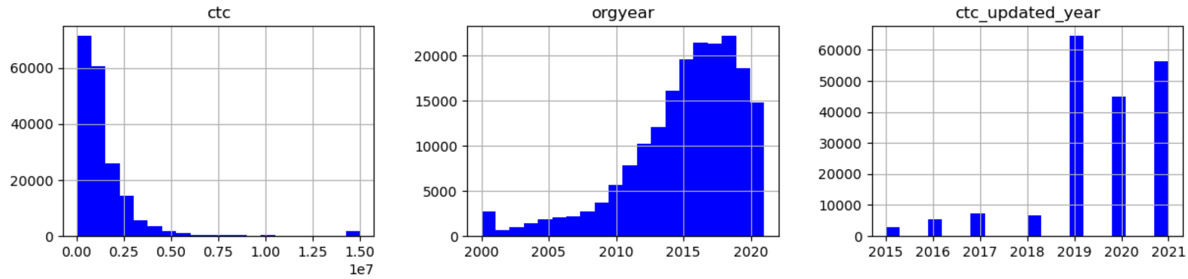
In [36]:

```
num_cols = ["ctc", "orgyear"]
fig , ax = plt.subplots(2,2,figsize=(15,7))
fig.set_facecolor("pink")
rows = 0
for col in num_cols:
    ax[rows][0].set_title("Boxplot for Outlier Detection ", fontweight="bold")
    plt.ylabel(col, fontsize=12)
    sns.boxplot(data = data_ctc_orgyear,y = data_ctc_orgyear[col],color='purple',ax
    # plt.subplot(nrows,mcols,pltcounter+1)
    sns.distplot(data_ctc_orgyear[col],color='purple',ax=ax[rows][1])
    ax[rows][1].axvline(data_ctc_orgyear[col].mean(), color='r', linestyle='--', la
    ax[rows][1].axvline(data_ctc_orgyear[col].median(), color='m', linestyle='-', l
    ax[rows][1].axvline(data_ctc_orgyear[col].mode()[0], color='royalblue', linesty
    ax[rows][1].set_title("Outlier Detection ", fontweight="bold")
    ax[rows][1].legend({'Mean':data_ctc_orgyear[col].mean(),'Median':data_ctc_orgye
                        'Mode':data_ctc_orgyear[col].mode()})
    rows += 1
plt.show()
```



In [37]:

```
data_ctc_orgyear["ctc_updated_year"] = data["ctc_updated_year"]

data_ctc_orgyear.hist(figsize = (20,10), bins = 20, layout = (3,4), color = 'blue')
plt.show()
```



In [38]:
```
data["clipped_ctc"] = data_ctc_orgyear["ctc"]
data["clipped_orgyear"] = data_ctc_orgyear["orgyear"]
```

In [39]:
```
data.head()
```

Out[39]:

	company_hash	orgyear	ctc	job_position	ctc_updated_year	clipped_ctc	clipped_orgyear
0	atrgxnnt xzaxv	2016.0	1100000	Other	2020.0	1100000	2016
1	qtrxvzwt xzegwgbb rxbxnta	2018.0	449999	FullStack Engineer	2019.0	449999	2018
2	ojzwnvwnxw vx	2015.0	2000000	Backend Engineer	2020.0	2000000	2015
3	ngpgutaxv	2017.0	700000	Backend Engineer	2019.0	700000	2017
4	qxen sqghu	2017.0	1400000	FullStack Engineer	2019.0	1400000	2017

Outlier detection and treatment

In [40]:
```
data.head()
```

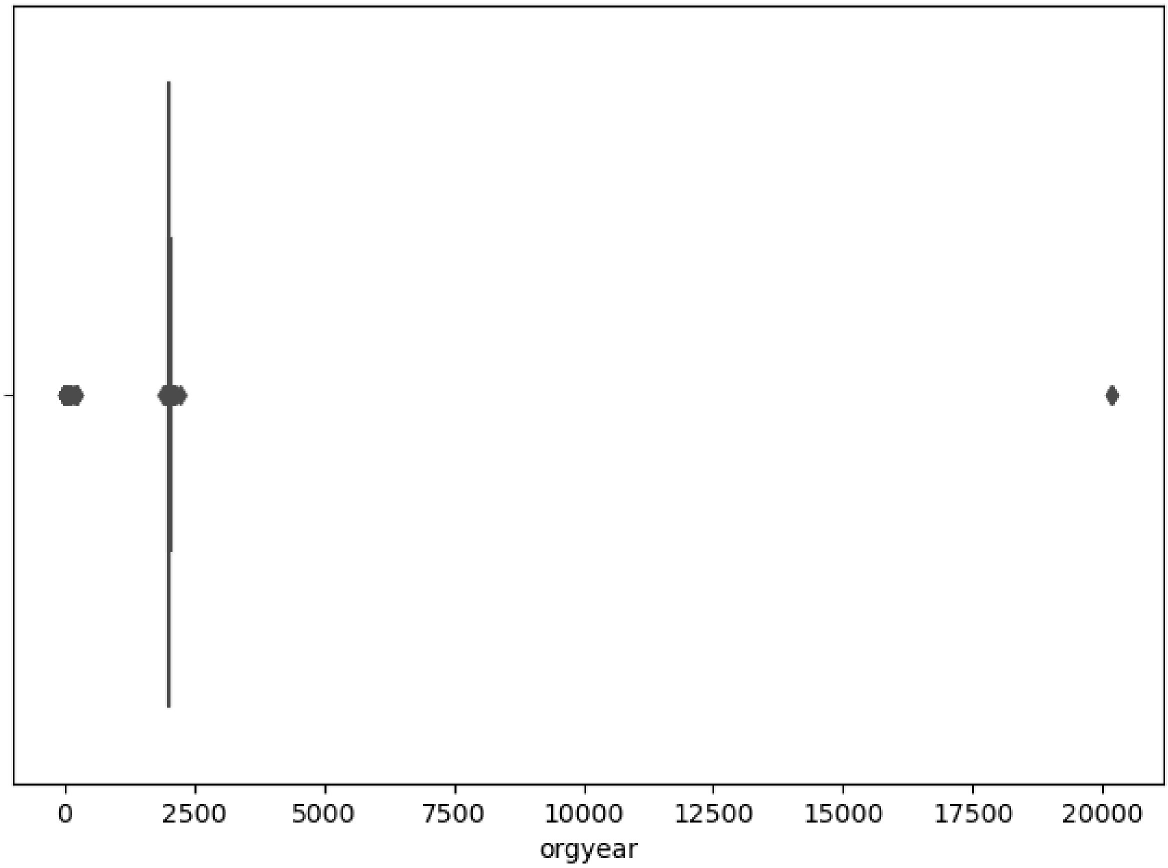
Out[40]:

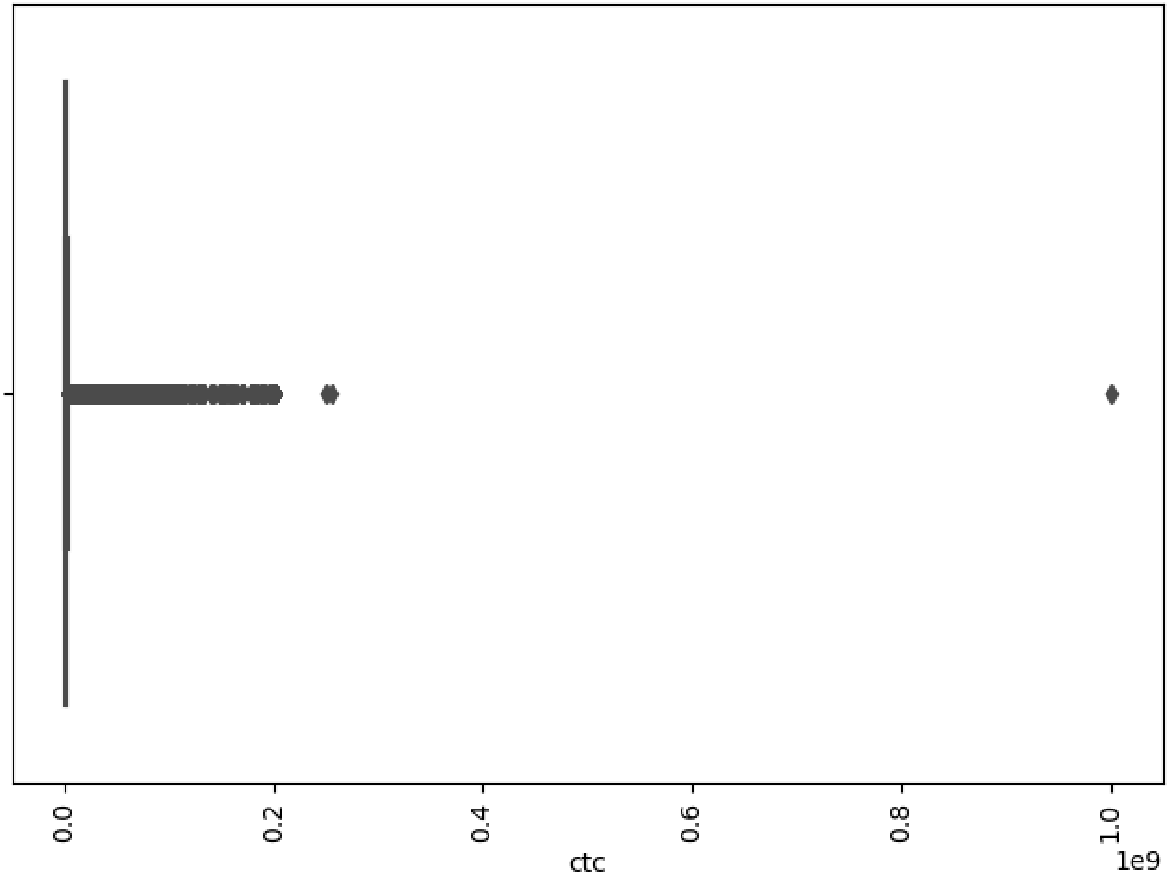
	company_hash	orgyear	ctc	job_position	ctc_updated_year	clipped_ctc	clipped_orgyear
0	atrgxnnt xzaxv	2016.0	1100000	Other	2020.0	1100000	2016
1	qtrxvzwt xzegwgbb rxbxnta	2018.0	449999	FullStack Engineer	2019.0	449999	2018
2	ojzwnvwnxw vx	2015.0	2000000	Backend Engineer	2020.0	2000000	2015
3	ngpgutaxv	2017.0	700000	Backend Engineer	2019.0	700000	2017
4	qxen sqghu	2017.0	1400000	FullStack Engineer	2019.0	1400000	2017

Check for Outliers

In [41]:
```
num_cols = ["orgyear", "ctc"]
fig, axis = plt.subplots(nrows=2, ncols=1, figsize=(8, 12))
index = 0
```

```
for row in range(2):
    sns.boxplot(data[num_cols[index]], ax=axis[row], palette="bright")
    index += 1
plt.xticks(rotation=90)
plt.show()

# Features like "ctc" and "orgyear" has many outliers. They need to be treated.
```

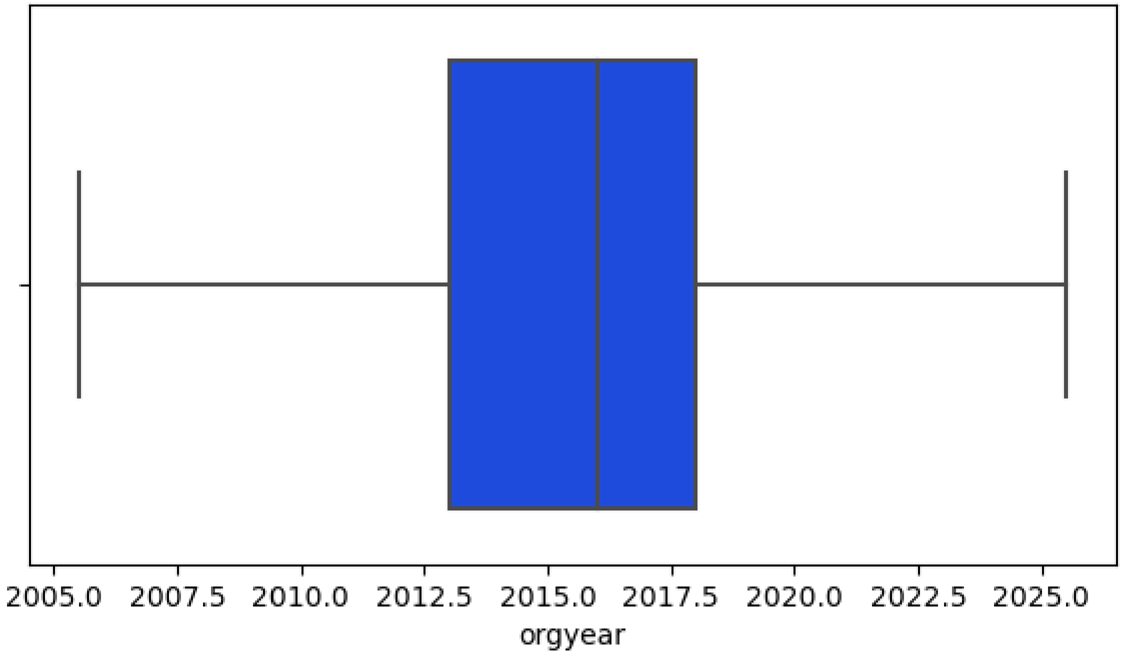


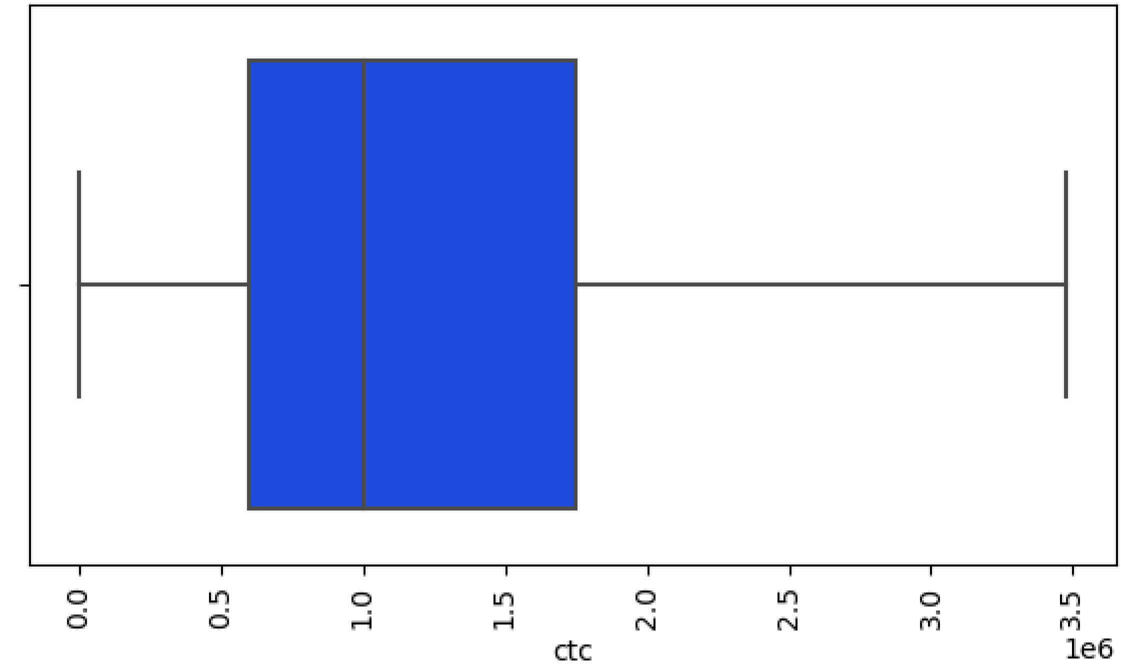


In [42]:

```
### Treatment of Outliers: Quantile based flooring and capping

fig, axis = plt.subplots(nrows=2, ncols=1, figsize=(7, 8))
index = 0
for row in range(2):
```

```
    q1 = np.percentile(data[num_cols[index]], 25)
    q3 = np.percentile(data[num_cols[index]], 75)
    IQR = q3-q1
    lower_bound = q1-(1.5*IQR)
    upper_bound = q3+(1.5*IQR)
    data[num_cols[index]] = np.where(data[num_cols[index]] < lower_bound, lower_bou
    data[num_cols[index]] = np.where(data[num_cols[index]] > upper_bound, upper_bou
    sns.boxplot(data[num_cols[index]], ax=axis[row], palette="bright")
    index += 1
plt.xticks(rotation=90)
plt.show()
```

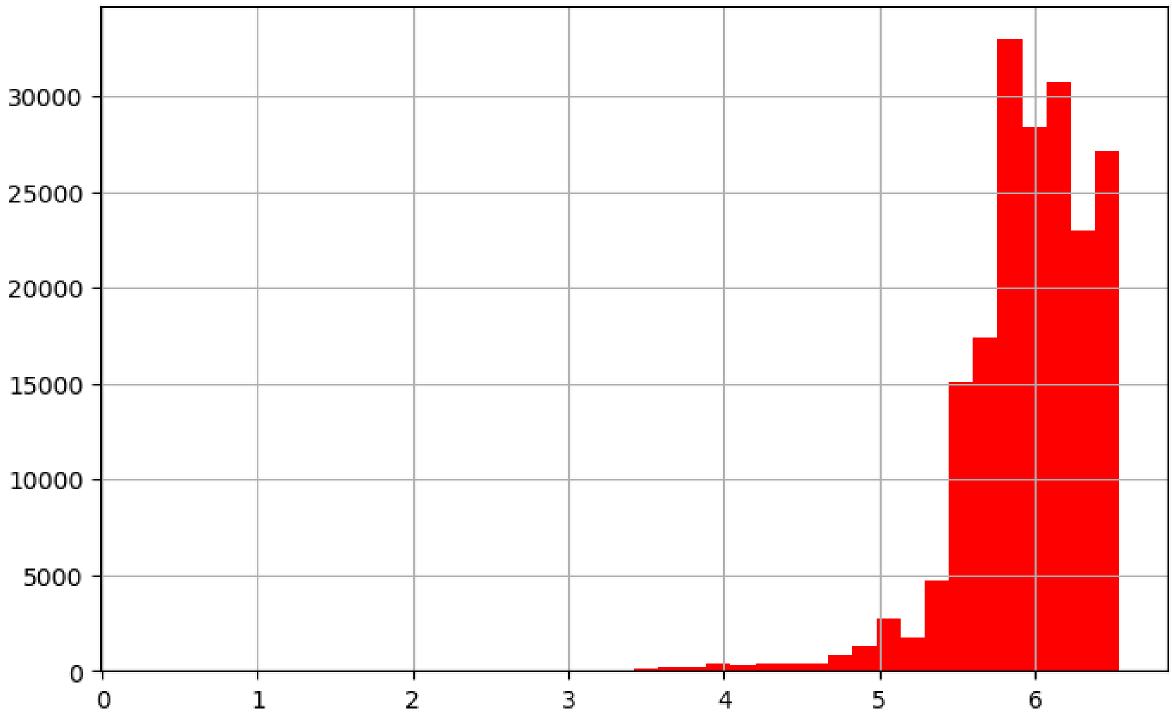




Feature Engineering

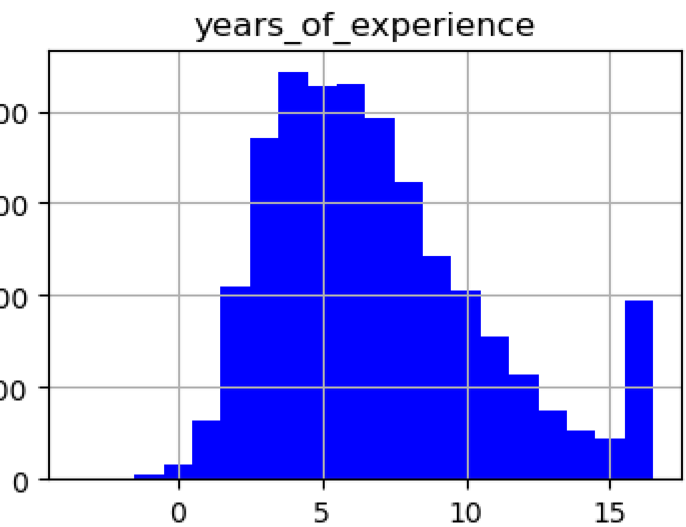
In [43]:
```
# creating new feature ctc_log_value
# as ctc values are very sparse in dataset, log values of ctc helps in calculation
data["ctc_log_value"] = np.log10(data['ctc'])
```

In [44]:
```
data["ctc_log_value"].hist(figsize = (8,5), bins = 40, color = "red")
plt.show()

# It can be observed that there are majority of learners with high ctc
```



In [45]:
```
# create new feature ‘Years of Experience’ column by subtracting orgyear from curre
data['years_of_experience']=2022-data['orgyear']
```

In [46]:
```
data[['years_of_experience']].hist(figsize = (20,10), bins = 20, layout = (3,4),col
plt.show()

# Majority of learners have 4-6 years of experience
```



In [47]:
```
#Creating flag indicating if a person got an increment or promotion or both in part

# Group the DataFrame by email and calculate the difference in ctc and job_position
data['ctc_change'] = data.groupby('company_hash')['ctc'].diff().fillna(0)
data['job_position_change'] = data.groupby('company_hash')['job_position'].apply(la

# Combine the flags to create a single flag indicating changes
data['inc_promotion_flag'] = 0
```

```
data.loc[data['ctc_change'] != 0, 'inc_promotion_flag'] += 1
data.loc[data['job_position_change'] != 0, 'inc_promotion_flag'] += 2

# Drop intermediate columns
data.drop(['ctc_change', 'job_position_change'], axis=1, inplace=True)
```

In [48]:

```
#Categorize ctc in high, low and average

# Define the bin edges and labels
bin_edges = [-float('inf'), 500000, 1000000, float('inf')]  # Define the bin edges
bin_labels = ['Low', 'Average', 'High']  # Define the bin labels

# Categorize 'CTC' into bins
data['CTC_category'] = pd.cut(data['ctc'], bins=bin_edges, labels=bin_labels)
data['CTC_category'].value_counts()

# Below are the counts of high, low and average CTCs.
```

Out[48]:

```
High       89766
Average    56994
Low        41527
Name: CTC_category, dtype: int64
```

In [49]:

```
# Calculate the average CTC per company
average_ctc_per_company = data.groupby('company_hash')['ctc'].mean().reset_index()

# Calculate the average CTC per job_position
average_ctc_per_job_position = data.groupby('job_position')['ctc'].mean().reset_inc

# Calculate the average CTC per year of experience
average_ctc_per_year_of_experience = data.groupby('years_of_experience')['ctc'].mea
```

In [50]:

```
print("Average CTC per Company:")
average_ctc_per_company
```

```
Average CTC per Company:
```

Out[50]:

	company_hash	ctc
0	0	100000.0
1	0000	300000.0
2	01 ojztqsj	550000.0
3	05mz exzytvrny uqxcvnt rxbxnta	1100000.0
4	1	175000.0
...	...	...
37270	zyvzwt wgzohrnxzs tzsxzttqo	940000.0
37271	zz	935000.0
37272	zzb ztdnstz vacxogqj ucn rna	600000.0
37273	zzgato	130000.0
37274	zzzbzb	720000.0

37275 rows × 2 columns

In [51]:

```
print("\nAverage CTC per Job Position:")
average_ctc_per_job_position
```

```
Average CTC per Job Position:
```

Out[51]:

	job_position	ctc
0		650000.0
1	SDE 2	1200000.0
2	7	445000.0
3	7033771951	3475000.0
4	737	350000.0
...	...	...
1001	student	1715000.0
1002	support escalation engineer	2000000.0
1003	system engineer	500000.0
1004	system software engineer	610000.0
1005	technology analyst	82000.0

1006 rows × 2 columns

In [52]:

```
print("\nAverage CTC per Year of Experience:")
average_ctc_per_year_of_experience
```

```
Average CTC per Year of Experience:
```

Out[52]:

	years_of_experience	ctc
0	-3.5	1.469667e+06
1	-3.0	6.802500e+05
2	-2.0	1.388488e+06
3	-1.0	1.282749e+06
4	0.0	1.258159e+06
5	1.0	1.061036e+06
6	2.0	1.016709e+06
7	3.0	9.823805e+05
8	4.0	1.006593e+06
9	5.0	1.046148e+06
10	6.0	1.134517e+06
11	7.0	1.203423e+06
12	8.0	1.291877e+06
13	9.0	1.420143e+06
14	10.0	1.546807e+06
15	11.0	1.653548e+06
16	12.0	1.758527e+06
17	13.0	1.824434e+06
18	14.0	1.867074e+06
19	15.0	1.978743e+06
20	16.0	2.075693e+06
21	16.5	2.228688e+06

In [53]:

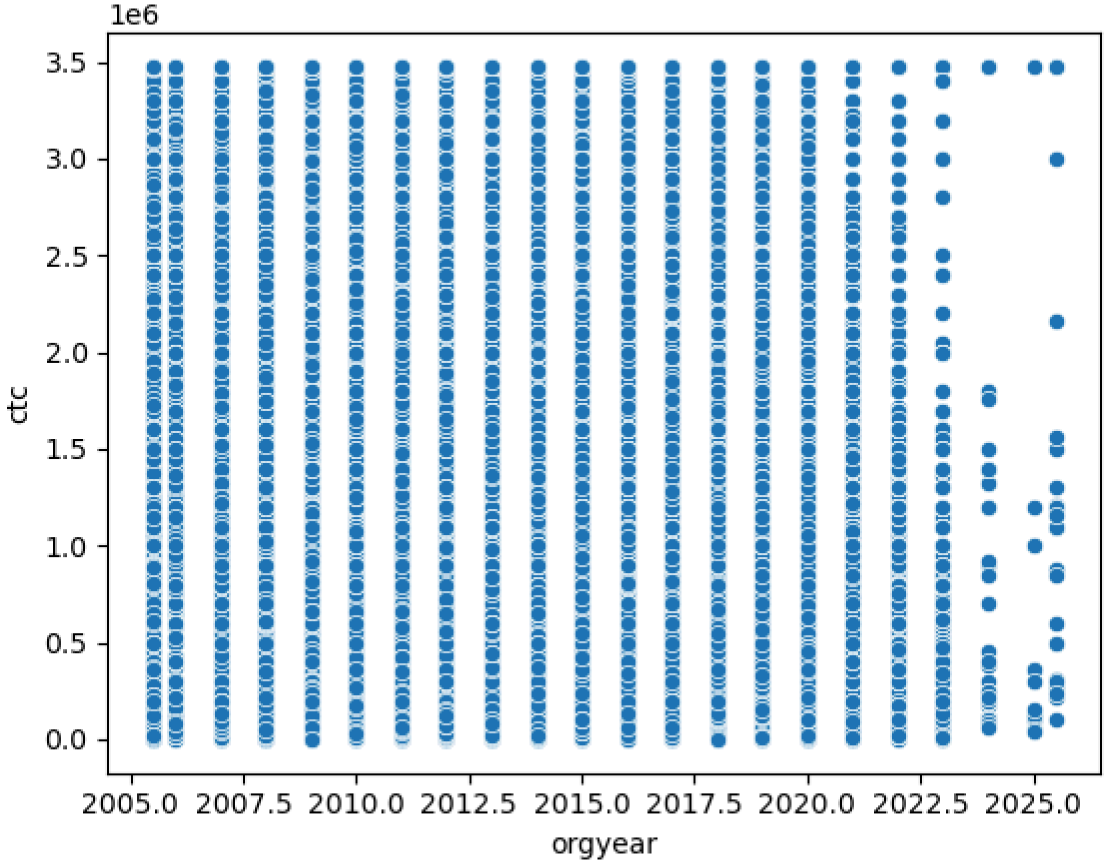
```
data.head()
```

Out[53]:

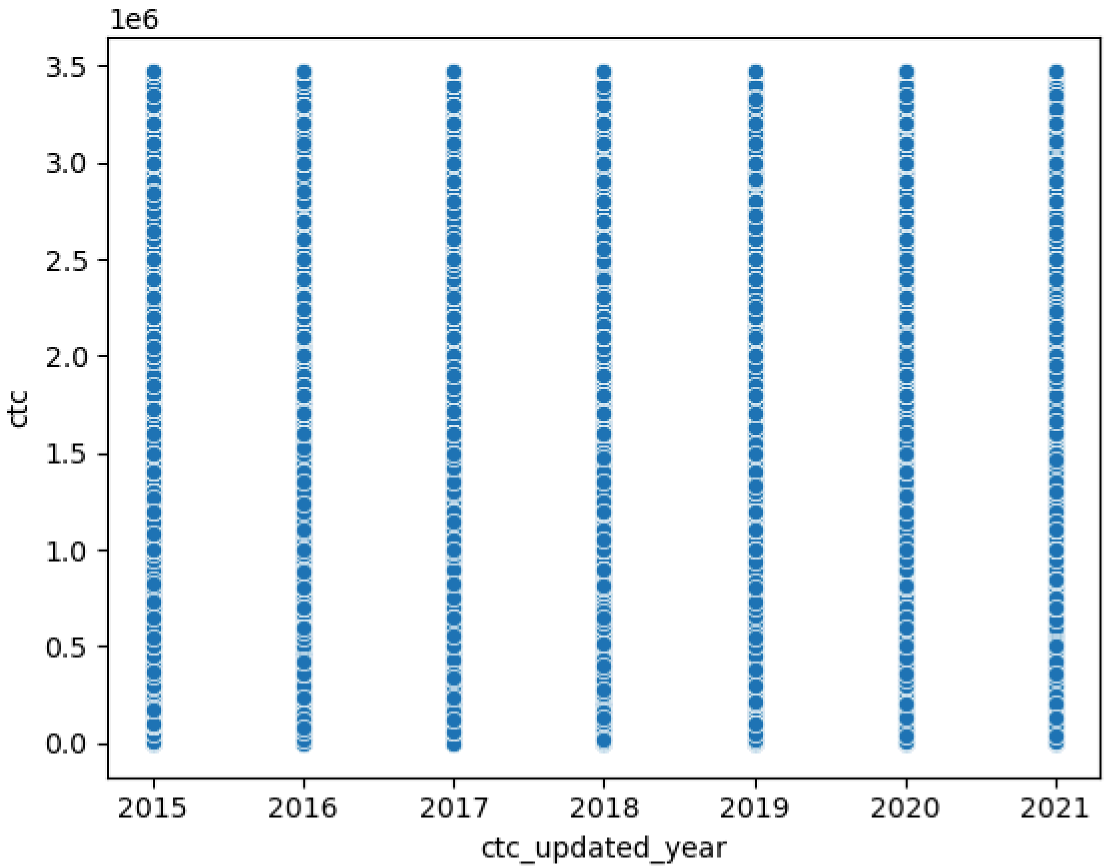
	company_hash	orgyear	ctc	job_position	ctc_updated_year	clipped_ctc	clipped_orgyear
0	atrgxnnt xzaxv	2016.0	1100000.0	Other	2020.0	1100000	2016
1	qtrxvzwt xzegwgbb rxbxnta	2018.0	449999.0	FullStack Engineer	2019.0	449999	2018
2	ojzwnvwnxw vx	2015.0	2000000.0	Backend Engineer	2020.0	2000000	2015
3	ngpgutaxv	2017.0	700000.0	Backend Engineer	2019.0	700000	2017
4	qxen sqghu	2017.0	1400000.0	FullStack Engineer	2019.0	1400000	2017

Bivariate Analysis

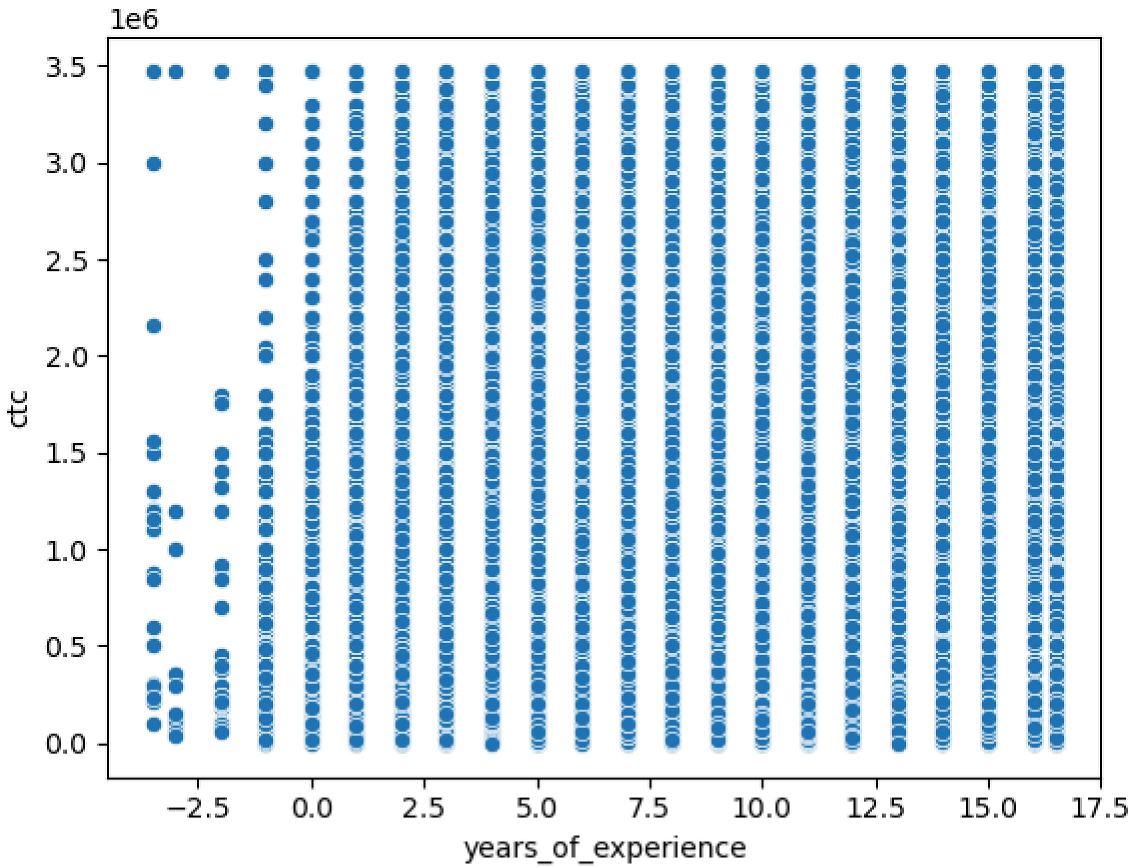
In [54]:
```
sns.scatterplot(data = data, x = "orgyear", y = "ctc")
plt.show()
```



In [55]:
```
sns.scatterplot(data = data, x = "ctc_updated_year", y = "ctc")
plt.show()
```

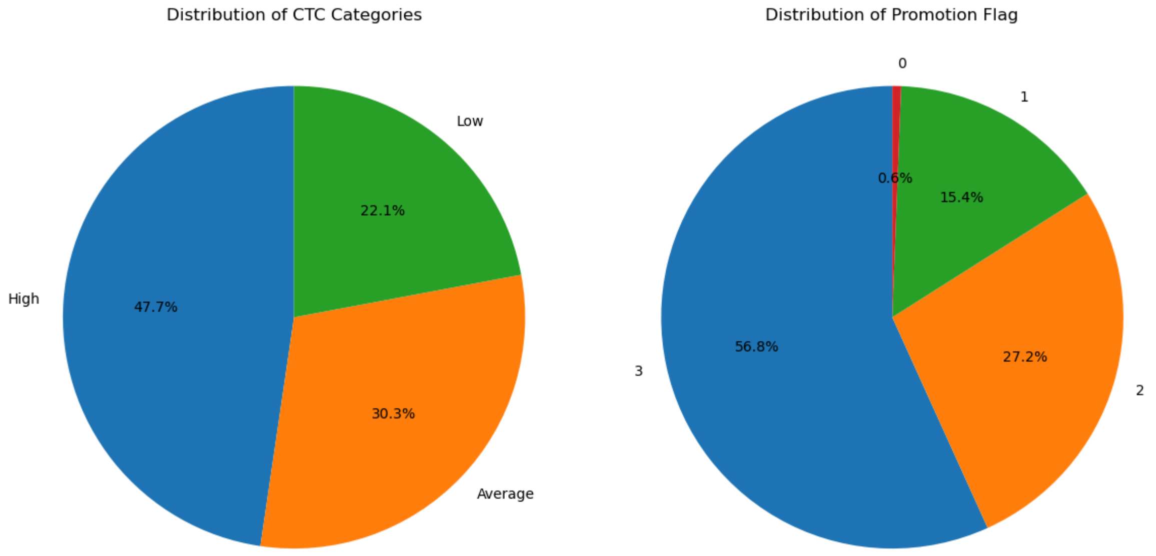


In [56]:
```
sns.scatterplot(data = data, x = "years_of_experience", y = "ctc")
plt.show()
```

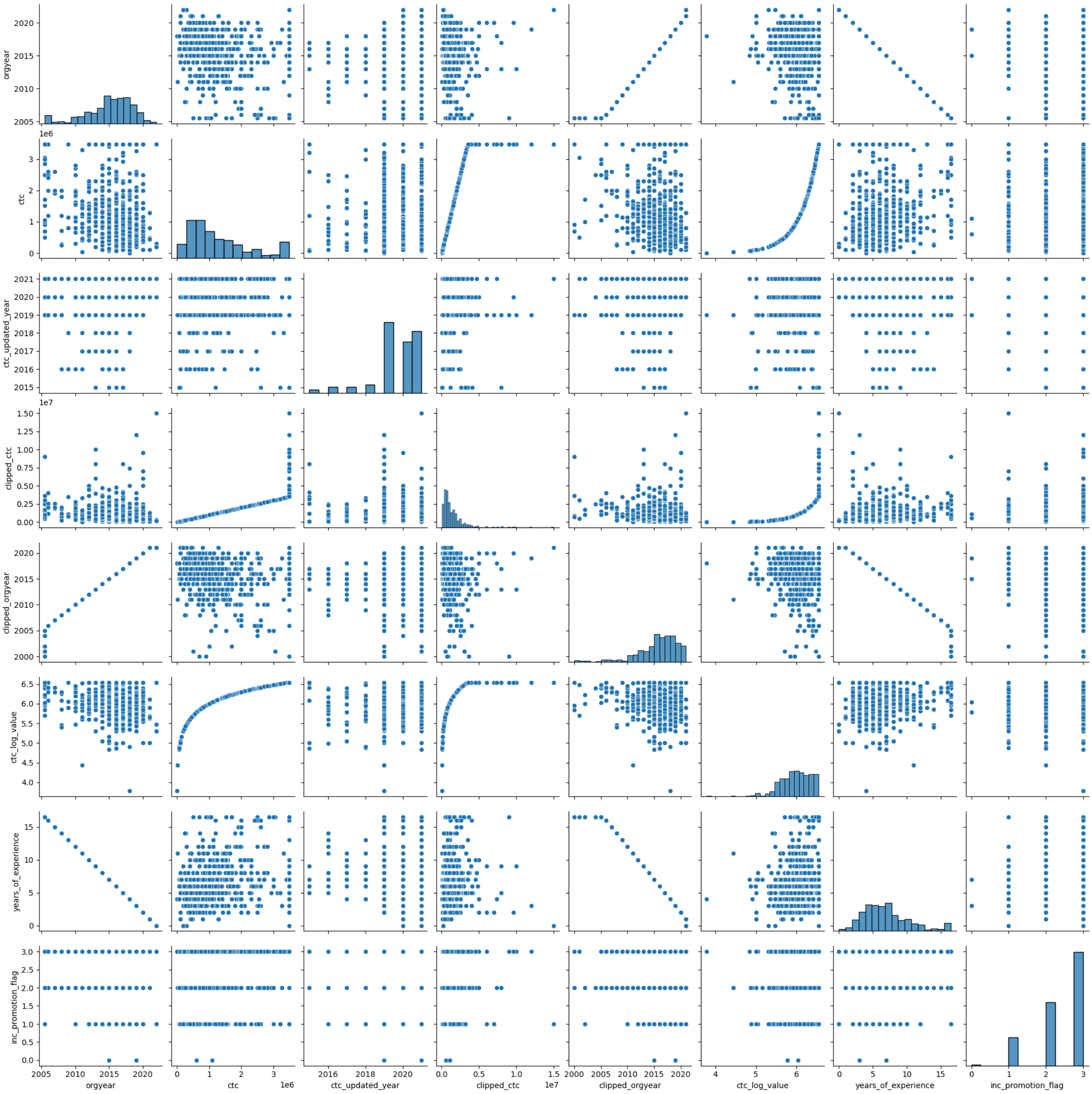


In [57]:
```
# Count the frequency of each category in 'CTC_category'
ctc_category_counts = data['CTC_category'].value_counts()

# Count the frequency of each category in 'inc_promotion_flag'
```

```
promotion_flag_counts = data['inc_promotion_flag'].value_counts()

# Create subplots for the pie charts
fig, axs = plt.subplots(1, 2, figsize=(12, 6))

# Plot the pie chart for 'CTC_category'
axs[0].pie(ctc_category_counts, labels=ctc_category_counts.index, autopct='%1.1f%%'
axs[0].set_title('Distribution of CTC Categories')

# Plot the pie chart for 'inc_promotion_flag'
axs[1].pie(promotion_flag_counts, labels=promotion_flag_counts.index, autopct='%1.1
axs[1].set_title('Distribution of Promotion Flag')

# Show the pie charts
plt.tight_layout()
plt.show()

# It can be seen that 47.7% learners are getting high paid salary, 30.3% earns aver
# 56.8% got both promotion and increment in ctc updated year, 27.2% got change job
```



In [58]:
```
sns.pairplot(data.sample(500))
plt.show()
```



Manual Clustering

Manual Clustering based on company, job position and years of experience

In [59]:
```
comp_job_year_group_data = data.groupby(['company_hash','job_position','years_of_ex
comp_job_year_group_data.head(2)
```

Out[59]:

			count	mean	std	min	25%	5
company_hash	**job_position**	**years_of_experience**						
0	**Other**	**2.0**	1.0	100000.0	NaN	100000.0	100000.0	10000
	nan	**2.0**	1.0	100000.0	NaN	100000.0	100000.0	10000

In [60]:
```
grouped_data_1 = data.merge(comp_job_year_group_data, on=['company_hash','job_posit
grouped_data_1.head(2)
```

Out[60]:

	company_hash	orgyear	ctc	job_position	ctc_updated_year	clipped_ctc	clipped_orgyear
0	atrgxnnt xzaxv	2016.0	1100000.0	Other	2020.0	1100000	2016
1	qtrxvzwt xzegwgbb rxbxnta	2018.0	449999.0	FullStack Engineer	2019.0	449999	2018

In [61]:
```
def flag(a,b_50,b_75):
    if a<b_50:
        return 3
    elif a>=b_50 and a<=b_75:
        return 2
    elif a>=b_75:
        return 1
```

In [62]:
```
grouped_data_1['designation'] = grouped_data_1.apply(lambda x: flag(x['ctc'],x['50%
grouped_data_1.head(2)
```

Out[62]:

	company_hash	orgyear	ctc	job_position	ctc_updated_year	clipped_ctc	clipped_orgyear
0	atrgxnnt xzaxv	2016.0	1100000.0	Other	2020.0	1100000	2016
1	qtrxvzwt xzegwgbb rxbxnta	2018.0	449999.0	FullStack Engineer	2019.0	449999	2018

In [63]:
```
data_1 = grouped_data_1.drop(columns = ["count","mean","std","min","25%","50%","75%
data_1.head(2)
```

Out[63]:

	company_hash	orgyear	ctc	job_position	ctc_updated_year	clipped_ctc	clipped_orgyear
0	atrgxnnt xzaxv	2016.0	1100000.0	Other	2020.0	1100000	2016
1	qtrxvzwt xzegwgbb rxbxnta	2018.0	449999.0	FullStack Engineer	2019.0	449999	2018

Manual Clustering based on company level and job_position level

In [64]:
```
comp_job_group_data = data_1.groupby(['company_hash','job_position'])['ctc'].descri
comp_job_group_data.head(2)
```

Out[64]:

		count	mean	std	min	25%	50%	75%	m
company_hash	job_position								
0	Other	1.0	100000.0	NaN	100000.0	100000.0	100000.0	100000.0	100000
	nan	1.0	100000.0	NaN	100000.0	100000.0	100000.0	100000.0	100000

In [65]:
```
grouped_data_2 = data_1.merge(comp_job_group_data, on=['company_hash','job_positior
grouped_data_2.head(2)
```

Out[65]:

	company_hash	orgyear	ctc	job_position	ctc_updated_year	clipped_ctc	clipped_orgyear
0	atrgxnnt xzaxv	2016.0	1100000.0	Other	2020.0	1100000	2016
1	qtrxvzwt xzegwgbb rxbxnta	2018.0	449999.0	FullStack Engineer	2019.0	449999	2018

In [66]:
```
grouped_data_2['class'] = grouped_data_2.apply(lambda x: flag(x['ctc'],x['50%'],x['
grouped_data_2.head(2)
```

Out[66]:

	company_hash	orgyear	ctc	job_position	ctc_updated_year	clipped_ctc	clipped_orgyear
0	atrgxnnt xzaxv	2016.0	1100000.0	Other	2020.0	1100000	2016
1	qtrxvzwt xzegwgbb rxbxnta	2018.0	449999.0	FullStack Engineer	2019.0	449999	2018

2 rows × 21 columns

In [67]:
```
data_2 = grouped_data_2.drop(columns = ["count","mean","std","min","25%","50%","75%
data_2.head(2)
```

Out[67]:

	company_hash	orgyear	ctc	job_position	ctc_updated_year	clipped_ctc	clipped_orgyear
0	atrgxnnt xzaxv	2016.0	1100000.0	Other	2020.0	1100000	2016
1	qtrxvzwt xzegwgbb rxbxnta	2018.0	449999.0	FullStack Engineer	2019.0	449999	2018

Manual Clustering based on company level

In [68]:
```
comp_group_data = data_2.groupby(['company_hash'])['ctc'].describe()
comp_group_data.head(2)
```

Out[68]:

	count	mean	std	min	25%	50%	75%	max
company_hash								
0	2.0	100000.0	0.0	100000.0	100000.0	100000.0	100000.0	100000.0
0000	1.0	300000.0	NaN	300000.0	300000.0	300000.0	300000.0	300000.0

In [69]:
```
grouped_data_3 = data_2.merge(comp_group_data, on=['company_hash'], how = 'left')
grouped_data_3.head(2)
```

Out[69]:

	company_hash	orgyear	ctc	job_position	ctc_updated_year	clipped_ctc	clipped_orgyear
0	atrgxnnt xzaxv	2016.0	1100000.0	Other	2020.0	1100000	2016
1	qtrxvzwt xzegwgbb rxbxnta	2018.0	449999.0	FullStack Engineer	2019.0	449999	2018

2 rows × 21 columns

In [70]:
```
grouped_data_3['tier'] = grouped_data_3.apply(lambda x: flag(x['ctc'],x['50%'],x['7
grouped_data_3.head(2)
```

Out[70]:

	company_hash	orgyear	ctc	job_position	ctc_updated_year	clipped_ctc	clipped_orgyear
0	atrgxnnt xzaxv	2016.0	1100000.0	Other	2020.0	1100000	2016
1	qtrxvzwt xzegwgbb rxbxnta	2018.0	449999.0	FullStack Engineer	2019.0	449999	2018

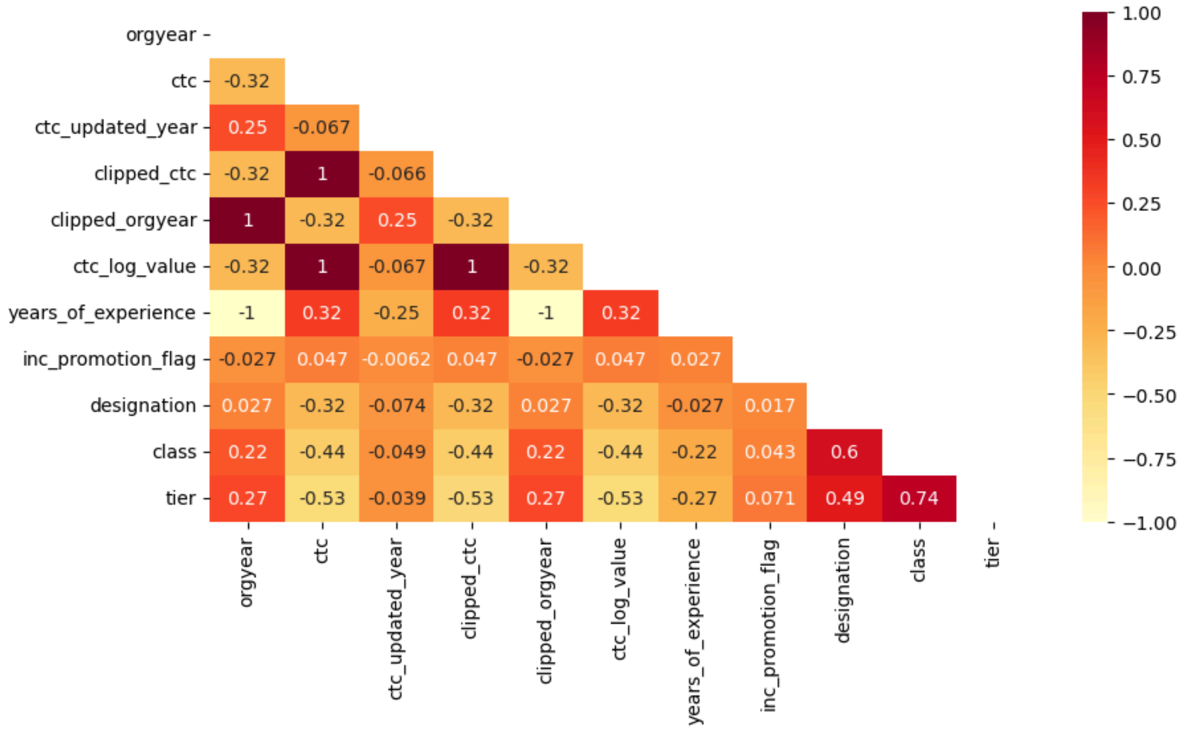
2 rows × 22 columns

In [71]:
```
data_3 = grouped_data_3.drop(columns = ["count","mean","std","min","25%","50%","75%
data_3.head(2)
```

Out[71]:

	company_hash	orgyear	ctc	job_position	ctc_updated_year	clipped_ctc	clipped_orgyear
0	atrgxnnt xzaxv	2016.0	1100000.0	Other	2020.0	1100000	2016
1	qtrxvzwt xzegwgbb rxbxnta	2018.0	449999.0	FullStack Engineer	2019.0	449999	2018

In [72]:
```
plt.figure(figsize=(10, 5))
corr = data_3.corr(method = 'spearman')
mask = np.triu(corr)
sns.heatmap(corr, annot = True,mask = mask, cmap = 'YlOrRd' )
plt.show()
```



Questions based on Manual Clustering

Top 10 employees (earning more than most of the employees in the company) - Tier 1

In [73]:
```
a1 = data_3.loc[data_3["tier"] == 1].sort_values(by = ["ctc_log_value"], ascending
a1
```

Out[73]:

	company_hash	orgyear	ctc	job_position	ctc_updated_year	clipped_ctc	clipped_o
188285	zgn vuurxwvmrt	2019.0	3475000.0	nan	2019.0	5100000	
114270	zgzt vn nyt bgbtzn	2011.0	3475000.0	Backend Engineer	2019.0	4000000	
114479	rgctrj uqgetooxgzvr hzxctqoxnj	2019.0	3475000.0	nan	2020.0	3500000	
114477	xb v onhatzn	2020.0	3475000.0	nan	2019.0	4800000	
114455	qvphntz	2019.0	3475000.0	FullStack Engineer	2019.0	5300000	
51701	xmb xzaxv uqxcvnt rxbxnta	2012.0	3475000.0	nan	2019.0	5300000	
51710	vnn	2016.0	3475000.0	Devops Engineer	2019.0	6000000	
51718	ovu	2006.0	3475000.0	Backend Architect	2019.0	3500000	
51751	vagmt	2013.0	3475000.0	QA Engineer	2019.0	4200000	
51755	vzvrjnxwo ihgnxtzn	2005.5	3475000.0	Engineering Leadership	2019.0	3850000	

Top 10 employees of data science in Amazon / TCS etc earning more than their peers - Class 1

In [74]:

```
a2 = data_3.loc[(data_3["class"] == 1) & (data_3["job_position"] == "Data Scientist

a2
```

Out[74]:

	company_hash	orgyear	ctc	job_position	ctc_updated_year	clipped_ctc	clipped_o
187499	gqvwrt	2015.0	3475000.0	Data Scientist	2020.0	3800000	
51604	ctqxkgz	2015.0	3475000.0	Data Scientist	2019.0	4800000	
127498	vagmt	2005.5	3475000.0	Data Scientist	2021.0	4500000	
126930	cgavegzt xatv	2009.0	3475000.0	Data Scientist	2019.0	5000000	
47545	owqj vzvrjnxwo	2017.0	3475000.0	Data Scientist	2016.0	8000000	
126809	wrhonq	2009.0	3475000.0	Data Scientist	2020.0	4900000	
126622	bh oxsbv mhoxztoo ogrhnxgzo	2010.0	3475000.0	Data Scientist	2019.0	3500000	
126388	wxowg	2012.0	3475000.0	Data Scientist	2019.0	5000000	
65720	xmb xzaxv uqxcvnt rxbxnta	2018.0	3475000.0	Data Scientist	2019.0	4200000	
125998	nvcvzn	2005.5	3475000.0	Data Scientist	2019.0	5000000	

Bottom 10 employees of data science in Amazon / TCS etc earning less than their peers - Class 3

In [75]:

```
a3 = data_3.loc[(data_3["class"] == 3) & (data_3["job_position"] == "Data Scientist

a3
```

Out[75]:

	company_hash	orgyear	ctc	job_position	ctc_updated_year	clipped_ctc	clipped_org
10521	srgmvrtast xzntrrxstzwt ge nyxzso	2017.0	4000.0	Data Scientist	2019.0	30000	
8493	bxyhu wgbbhzxwvnxgz	2018.0	4000.0	Data Scientist	2019.0	30000	
47560	onhatzn	2021.0	6000.0	Data Scientist	2019.0	30000	
125837	ovbohzs trtwnqg btwyvzxwo	2017.0	7000.0	Data Scientist	2019.0	30000	
22959	exznqhon ogrhnxgzo ucn rna	2017.0	7200.0	Data Scientist	2019.0	30000	
9164	nvnv wgzohrnvzwj otqcxwto	2020.0	7500.0	Data Scientist	2020.0	30000	
22956	vqxosrgmvr	2015.0	8800.0	Data Scientist	2019.0	30000	
29972	sggsrt	2018.0	10000.0	Data Scientist	2021.0	30000	
153978	ytfrtnn uvwpvqa tzntquqxot	2018.0	10000.0	Data Scientist	2019.0	30000	
76735	uvjovet sqghu	2018.0	10000.0	Data Scientist	2019.0	30000	

Bottom 10 employees (earning less than most of the employees in the company)- Tier 3

In [76]:

```
a4 = data_3.loc[data_3["tier"] == 3].sort_values(by = ["ctc_log_value"], ascending
a4
```

Out[76]:

	company_hash	orgyear	ctc	job_position	ctc_updated_year	clipped_ctc	clipped_orgye
124374	xzntqcxtfmxn	2014.0	2.0	Backend Engineer	2019.0	30000	20
108696	xzntqcxtfmxn	2013.0	6.0	nan	2018.0	30000	20
105028	xzntqcxtfmxn	2013.0	14.0	nan	2018.0	30000	20
169294	xm	2016.0	15.0	nan	2018.0	30000	20
107562	hzxctqoxnj ge fvoyxzsngz	2022.0	200.0	nan	2021.0	30000	20
156708	nvnv wgzohrnvzwj otqcxwto	2012.0	600.0	Backend Engineer	2017.0	30000	20
138154	zvz	2023.0	600.0	nan	2019.0	30000	20
91916	gjg	2018.0	600.0	FullStack Engineer	2021.0	30000	20
57601	sttpoegqsttpo	2016.0	1000.0	nan	2019.0	30000	20
146185	ygbt atugn	2005.5	1000.0	FullStack Engineer	2018.0	30000	20

Top 10 employees in Amazon- X department - having 5/6/7 years of experience earning more than their peers - Tier X

In [77]:

```
# As company names in dataset has been masked, data for Amazon cannot be retrieved.
# Therefore, calculating only for employees having 5/6/7 years of experience in res

# Step 1: Filter employees with 5, 6, or 7 years of experience
filtered_df = data_3[data_3['years_of_experience'].isin([5, 6, 7])]

# Step 2: Filter employees in Tier X
filtered_df_1 = filtered_df[filtered_df['tier'] == 1]
filtered_df_2 = filtered_df[filtered_df['tier'] == 2]
filtered_df_3 = filtered_df[filtered_df['tier'] == 3]

# Step 3: Sort the DataFrame by earnings (ctc)
sorted_df_1 = filtered_df_1.sort_values(by='ctc_log_value', ascending=False)
sorted_df_2 = filtered_df_2.sort_values(by='ctc_log_value', ascending=False)
sorted_df_3 = filtered_df_3.sort_values(by='ctc_log_value', ascending=False)

# Step 4: Select the top 10 employees
top_10_employees_tier1 = sorted_df_1.head(10)
top_10_employees_tier2 = sorted_df_2.head(10)
top_10_employees_tier3 = sorted_df_3.head(10)
```

In [78]:

```
# Display the result
top_10_employees_tier1
```

Out[78]:

	company_hash	orgyear	ctc	job_position	ctc_updated_year	clipped_ctc	clipped_o
188272	vbvkgz	2016.0	3475000.0	nan	2020.0	4800000	
136021	nvnv wgzohrnvzwj otqcxwto rxbxnta	2017.0	3475000.0	Other	2021.0	15000000	
136758	uyvqbtvoj	2015.0	3475000.0	Engineering Leadership	2020.0	3750000	
136750	vbvatho xn sqghu	2015.0	3475000.0	FullStack Engineer	2020.0	5000000	
136384	vbvkgz	2017.0	3475000.0	FullStack Engineer	2021.0	4430000	
41869	xzegojo	2015.0	3475000.0	nan	2019.0	15000000	
136245	vbvkgz	2016.0	3475000.0	Backend Engineer	2021.0	14000000	
136160	vbvkgz	2017.0	3475000.0	FullStack Engineer	2021.0	4500000	
135989	vbvkgz	2017.0	3475000.0	Backend Engineer	2019.0	4400000	
41790	dtqgd	2017.0	3475000.0	Android Engineer	2020.0	15000000	

In [79]:

```
top_10_employees_tier2
```

Out[79]:

	company_hash	orgyear	ctc	job_position	ctc_updated_year	clipped_ctc	clipped_o
150123	ohbg rgsxw	2017.0	3475000.0	Backend Engineer	2019.0	3480000	
156690	ouqxzprq	2017.0	3475000.0	Engineering Leadership	2019.0	4000000	
22667	ygnonvq	2016.0	3475000.0	Backend Engineer	2020.0	4000000	
142098	vutdvx	2017.0	3475000.0	QA Engineer	2019.0	8000000	
178566	oxzsvugqt tdwyvzst	2015.0	3475000.0	Backend Engineer	2019.0	9000000	
14737	aqtvb11 sqghu ge wgbuvzxto	2015.0	3475000.0	Backend Engineer	2021.0	3500000	
101108	bxwqgogen	2015.0	3475000.0	nan	2021.0	6200000	
95177	zthmtqstq mtqbvz	2017.0	3475000.0	Support Engineer	2020.0	7000000	
178686	bxwqgogen	2015.0	3475000.0	Frontend Engineer	2020.0	3500000	
27486	ztnwrgha ojontbo uqxcvnt rxbxnta	2015.0	3475000.0	Android Engineer	2020.0	15000000	

In [80]:

```
display(top_10_employees_tier3)
```

	company_hash	orgyear	ctc	job_position	ctc_updated_year	clipped_ctc	clipped_o
167375	hmtq	2017.0	3400000.0	Backend Engineer	2020.0	3400000	
2813	wrghatqv	2016.0	3400000.0	nan	2021.0	3400000	
12505	wrghatqv	2016.0	3400000.0	Backend Engineer	2021.0	3400000	
143767	urvntxi	2016.0	3400000.0	FullStack Engineer	2021.0	3400000	
183151	gqvwrt wrgha	2016.0	3300000.0	Backend Engineer	2020.0	3300000	
148296	qhmqxp xzw	2017.0	3300000.0	Backend Engineer	2019.0	3300000	
44618	hmtq	2016.0	3300000.0	Backend Engineer	2021.0	3300000	
154456	xzattawgb	2017.0	3300000.0	nan	2019.0	3300000	
131042	xzattawgb	2017.0	3300000.0	Product Manager	2019.0	3300000	
116117	wgatzvnxgz	2017.0	3260000.0	FullStack Engineer	2016.0	3260000	

Top 10 companies (based on their CTC)

In [81]:
```
a5 = data_3.groupby(by = "company_hash")["ctc"].mean().round(2).reset_index().sort_

a5
```

Out[81]:

	company_hash	ctc
16785	ovbmv nc	3475000.0
27009	vhqxsg ntwyzgrgsxto	3475000.0
5408	egqfvqa vxa eghzavnxgz	3475000.0
7989	gvzav xzaxv	3475000.0
1186	auo ntrtwgb	3475000.0
33297	xwh btaxwvr qa ogenfvqt atcxwto uexktq wytzzvx	3475000.0
12931	ntqhbg ytvqn xzw	3475000.0
25135	uqxgqxnj ogenfvqt	3475000.0
8753	hzxnta onvnto ongct wgbuvzj	3475000.0
32973	xnxav	3475000.0

Top 2 positions in every company (based on their CTC)

In [82]:
```
a6 = pd.pivot_table(data = data_3,values = 'ctc', index = ['company_hash', 'job_pos
                    fill_value = None, margins = False,dropna = True, sort = True).r
a6 = a6.groupby(by = ['company_hash']).apply(lambda x : x.sort_values(by = 'ctc', a
a6 = a6.reset_index(drop = True)
a6['ctc'] = a6['ctc'].astype('int')
a6
```

Out[82]:

	index	company_hash	job_position	ctc
0	0	0	Other	100000
1	1	0	nan	100000
2	2	0000	Other	300000
3	4	01 ojztqsj	Frontend Engineer	830000
4	3	01 ojztqsj	Android Engineer	270000
...	...	...	...	...
50224	71256	zz	nan	500000
50225	71257	zzb ztdnstz vacxogqj ucn rna	FullStack Engineer	600000
50226	71258	zzb ztdnstz vacxogqj ucn rna	nan	600000
50227	71259	zzgato	nan	130000
50228	71260	zzzbzb	Other	720000

50229 rows × 4 columns

In [83]:
```
data_no_std = data_3.copy()
```

Data processing for Unsupervised clustering - Label encoding/ One- hot encoding, Standardization of data

In [84]:
```
df = data_3
```

LabelEncoding of some features

In [85]:
```
from sklearn import preprocessing
label_encoder = preprocessing.LabelEncoder()
```

In [86]:
```
df['job_position']= label_encoder.fit_transform(df['job_position'])
```

In [87]:
```
df['company_hash']= label_encoder.fit_transform(df['company_hash'])
```

In [88]:
```
df['CTC_category']= label_encoder.fit_transform(df['CTC_category'])
```

In [89]:
```
df.head()

# orgyear can be deleted as years of experience reflect same data
# Similarly, ctc_updated_year can also be deleted.
```

Out[89]:

	company_hash	orgyear	ctc	job_position	ctc_updated_year	clipped_ctc	clipped_orgyear
0	968	2016.0	1100000.0	455	2020.0	1100000	2016
1	19712	2018.0	449999.0	289	2019.0	449999	2018
2	15499	2015.0	2000000.0	138	2020.0	2000000	2015
3	12100	2017.0	700000.0	138	2019.0	700000	2017
4	20208	2017.0	1400000.0	289	2019.0	1400000	2017

In [90]:
```
df.drop(columns=['orgyear'],inplace=True)
df.drop(columns=['ctc_updated_year'],inplace=True)
df.drop(columns=['clipped_ctc'],inplace=True)
df.drop(columns=['clipped_orgyear'],inplace=True)
```

In [91]:
```
df.head()
```

Out[91]:

	company_hash	ctc	job_position	ctc_log_value	years_of_experience	inc_promotion_flag
0	968	1100000.0	455	6.041393	6.0	2
1	19712	449999.0	289	5.653212	4.0	2
2	15499	2000000.0	138	6.301030	7.0	2
3	12100	700000.0	138	5.845098	5.0	2
4	20208	1400000.0	289	6.146128	5.0	2

Standardization of data - Standard Scaling

In [92]:
```
from sklearn.preprocessing import StandardScaler

scale = StandardScaler()

X = scale.fit_transform(df)
X = pd.DataFrame(X, columns = df.columns, index = df.index)
```

Unsupervised Learning - Clustering

Checking clustering tendency

In [93]:
```
from sklearn.cluster import KMeans
```

In [94]:
```
k = 4 ## randomly chosen value value
kmeans = KMeans(n_clusters=k)
y_pred = kmeans.fit_predict(X)
```

In [95]:
```
kmeans.cluster_centers_
```

Out[95]:
```
array([[ 0.09859047, -1.01680418,  0.12945275, -1.36355376, -0.36597388,
        -0.05478621,  1.50574062,  0.48200863,  0.57274081,  0.63045219],
       [-0.06788149,  1.33007762, -0.1594254 ,  1.01231286,  0.72055921,
         0.0975277 ,  0.112516  , -0.53200768, -0.87713097, -1.05816178],
       [ 0.04798315, -0.23934837,  0.10700274,  0.07736753, -0.20563061,
        -0.24688303, -0.66623423, -0.46253385, -0.42918598, -0.3161737 ],
       [-0.06857266, -0.23820694, -0.06443452,  0.05457227, -0.20894841,
         0.21770285, -0.71700949,  0.64370657,  0.8903809 ,  0.90557905]])
```

In [96]:
```
kmeans.labels_
```

Out[96]:
```
array([2, 0, 1, ..., 3, 1, 3])
```

In [97]:
```
clusters = pd.DataFrame(X, columns=df.columns)
clusters['label'] = kmeans.labels_
clusters
```

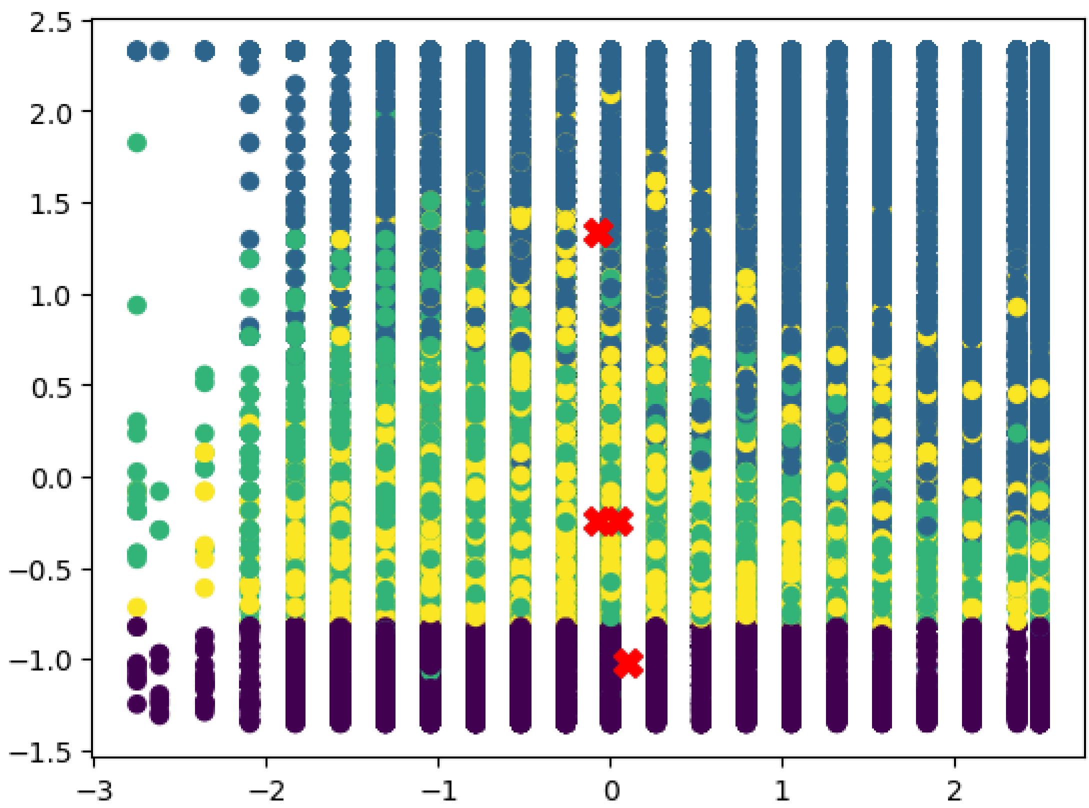
Out[97]:

	company_hash	ctc	job_position	ctc_log_value	years_of_experience	inc_promotion_
0	-1.643699	-0.188928	-0.009523	0.190035	-0.257473	-0.524
1	0.028677	-0.877417	-0.507788	-0.747972	-0.782065	-0.524
2	-0.347215	0.764363	-0.961029	0.817427	0.004823	-0.524
3	-0.650480	-0.612613	-0.961029	-0.284294	-0.519769	-0.524
4	0.072931	0.128835	-0.507788	0.443119	-0.519769	-0.524
...	...	...	...	...	...	
188282	0.833370	-1.121035	1.572318	-1.498967	1.840894	-1.831
188283	-0.971501	-0.824455	1.572318	-0.637400	-0.519769	0.782
188284	0.863081	-0.612613	1.572318	-0.284294	-1.568953	-1.831
188285	1.482460	2.326700	1.572318	1.397185	-1.044361	-1.831
188286	-1.536811	-0.040638	1.572318	0.315759	0.267119	-1.831

188287 rows × 11 columns

In [98]:

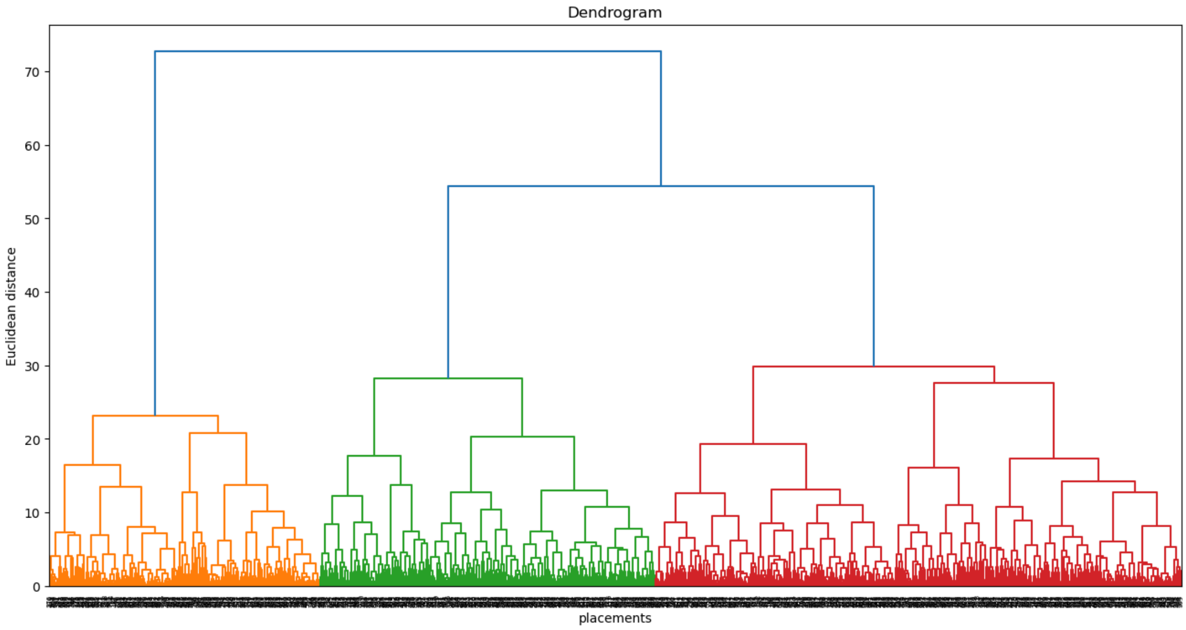
```
plt.scatter(clusters['years_of_experience'], clusters['ctc'], c=clusters['label'])
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], color="re
plt.show()
```



In [99]:

```
from scipy.cluster.hierarchy import dendrogram
import scipy.cluster.hierarchy as sch

plt.figure(figsize = (16,8))

dendrogrm = sch.dendrogram(sch.linkage(X.sample(1000), method = 'ward'))
plt.title('Dendrogram')
```

```
plt.xlabel('placements')
plt.ylabel('Euclidean distance')
plt.show()

# On observing dendogram, the best possible clusters formation will be 4.
# Check with Elbow Method
```



Elbow method

In [100…

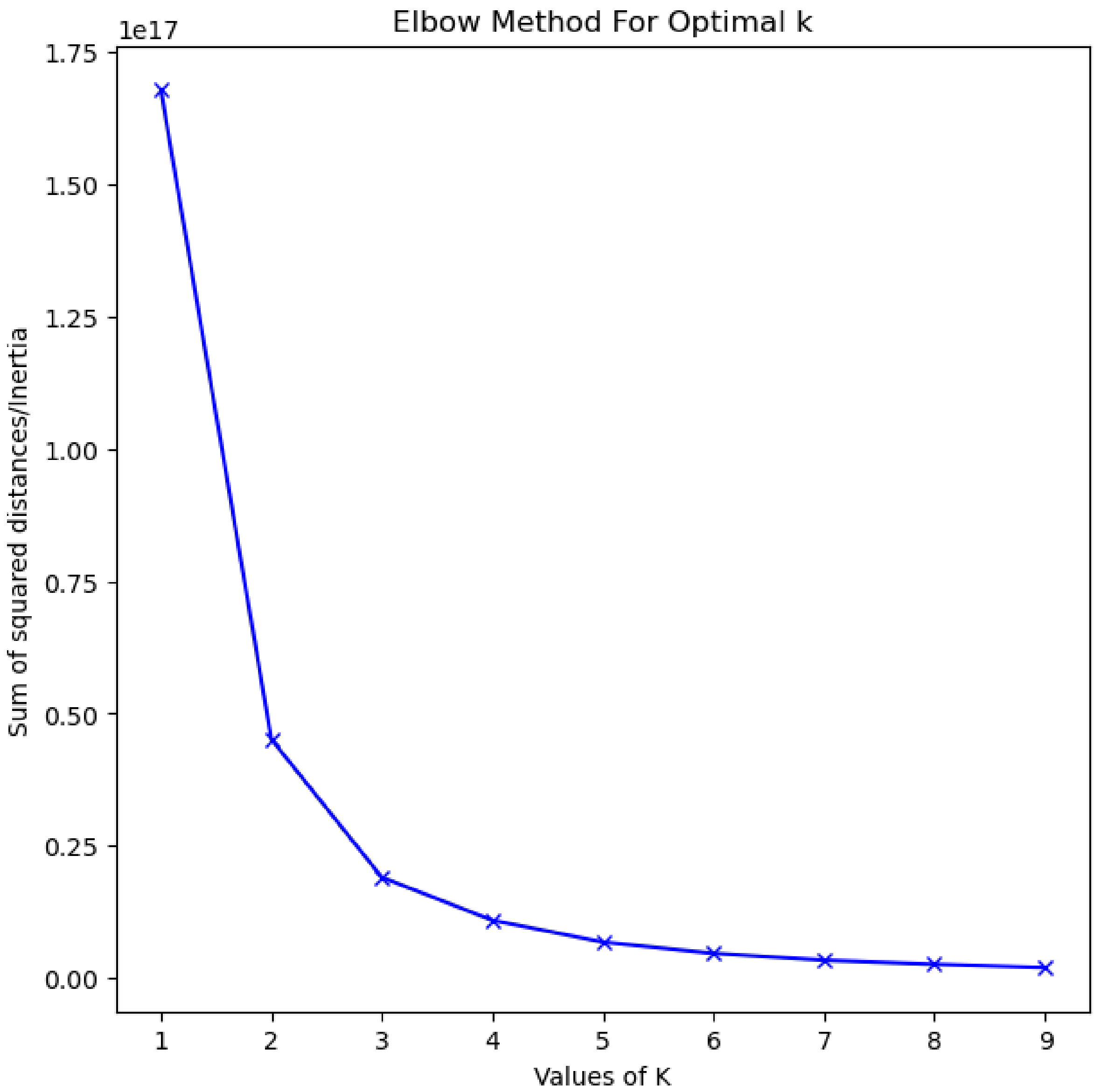
```
df_1 = df[["company_hash", "ctc","job_position", "years_of_experience","designatior
df_1.head()
```

Out[100]:

	company_hash	ctc	job_position	years_of_experience	designation	class	tier
0	968	1100000.0	455	6.0	2	1	2
1	19712	449999.0	289	4.0	3	3	3
2	15499	2000000.0	138	7.0	2	2	2
3	12100	700000.0	138	5.0	3	3	3
4	20208	1400000.0	289	5.0	2	1	1

In [101…

```
# Use Elbow method to choose best k

plt.figure(figsize = (7,7))
Sum_of_squared_distances = []
K = range(1,10)
for num_clusters in K :
    kmeans = KMeans(n_clusters=num_clusters)
    kmeans.fit(df_1)
    Sum_of_squared_distances.append(kmeans.inertia_)
plt.plot(K,Sum_of_squared_distances,'bx-')
plt.xlabel('Values of K')
plt.ylabel('Sum of squared distances/Inertia')
plt.title('Elbow Method For Optimal k')
plt.show()
```

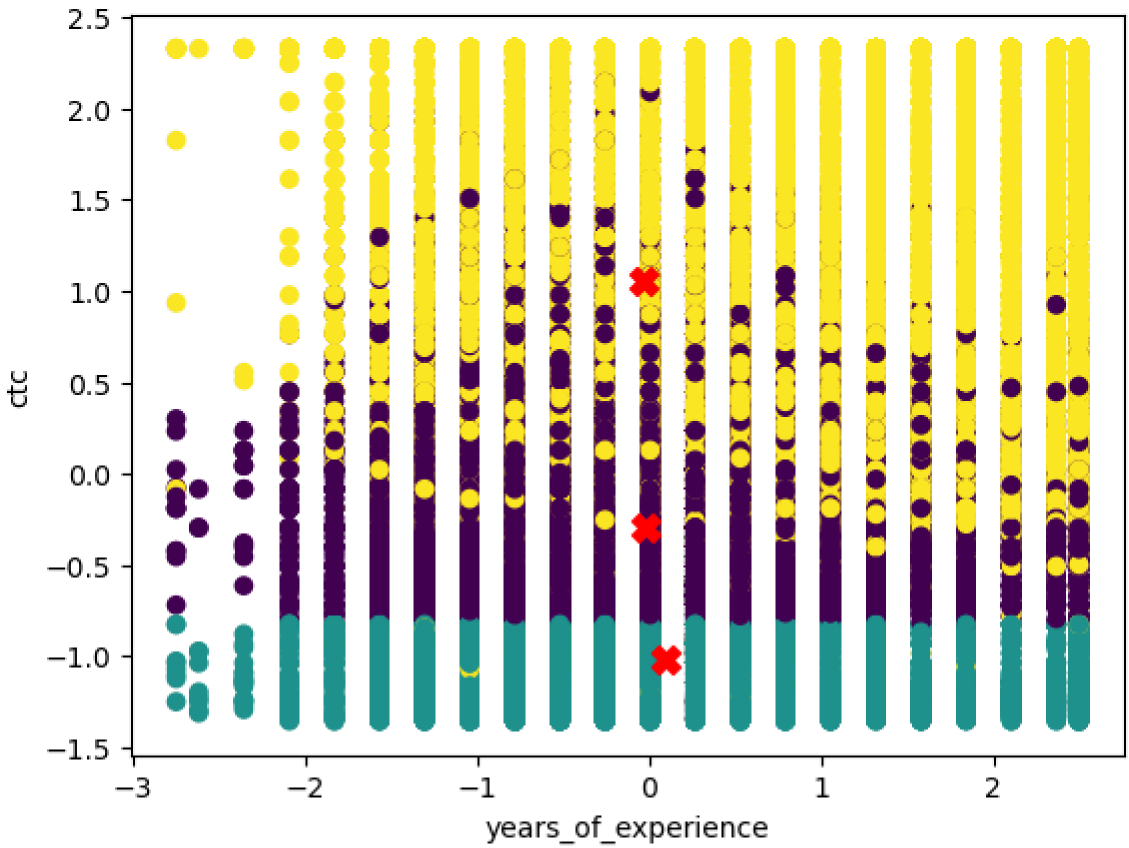


K = 3 or 4 is best choice for kmeans clustering. Already seen for 4, let's try for k = 3

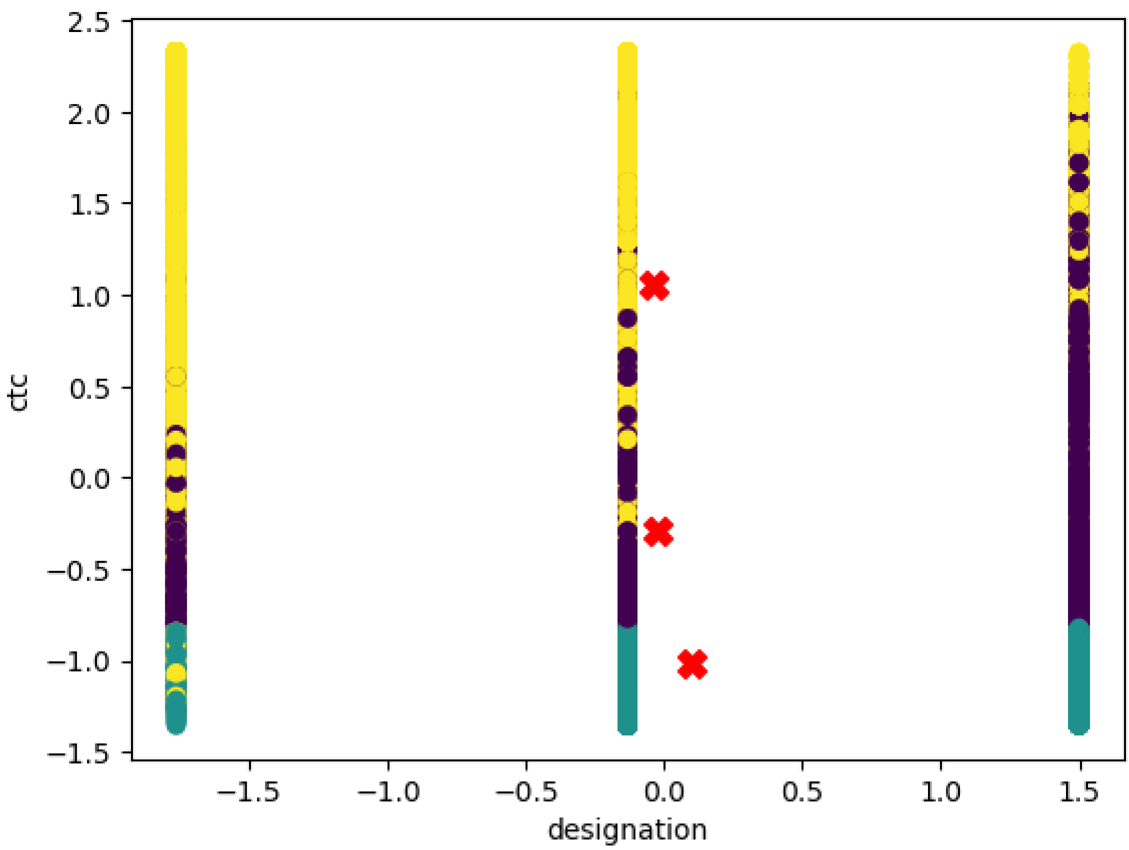
K-means clustering

In [102…

```
k = 3 ## chosen k using elbow method
kmeans = KMeans(n_clusters=k)
y_pred = kmeans.fit_predict(X)

clusters = pd.DataFrame(X, columns=df.columns)
clusters['label'] = kmeans.labels_

plt.scatter(clusters['years_of_experience'], clusters['ctc'], c=clusters['label'])
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], color="re
plt.xlabel("years_of_experience")
plt.ylabel("ctc")
plt.show()

# There is not enough clarity for k = 4 as observed before, Therefore, choose k = 3
```

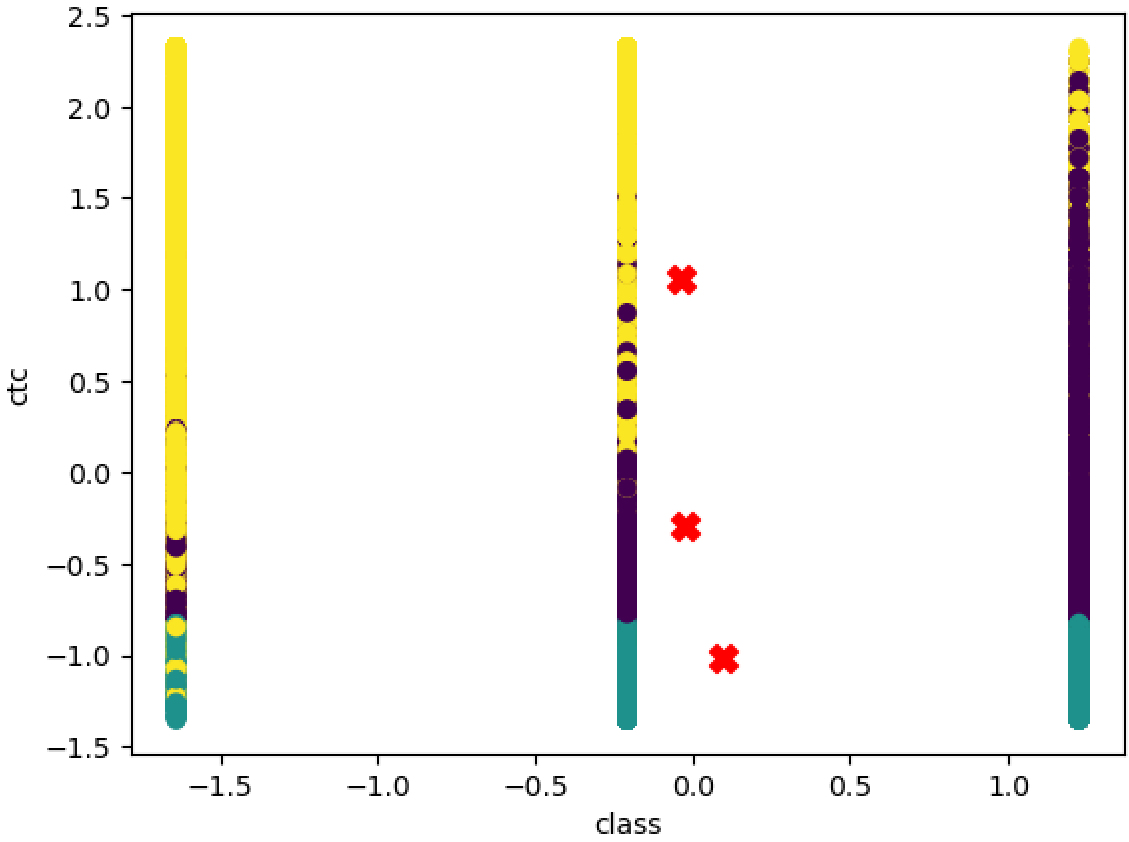


In [103…
```
plt.scatter(clusters['designation'], clusters['ctc'], c=clusters['label'])
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], color="re
plt.xlabel("designation")
plt.ylabel("ctc")
plt.show()
```



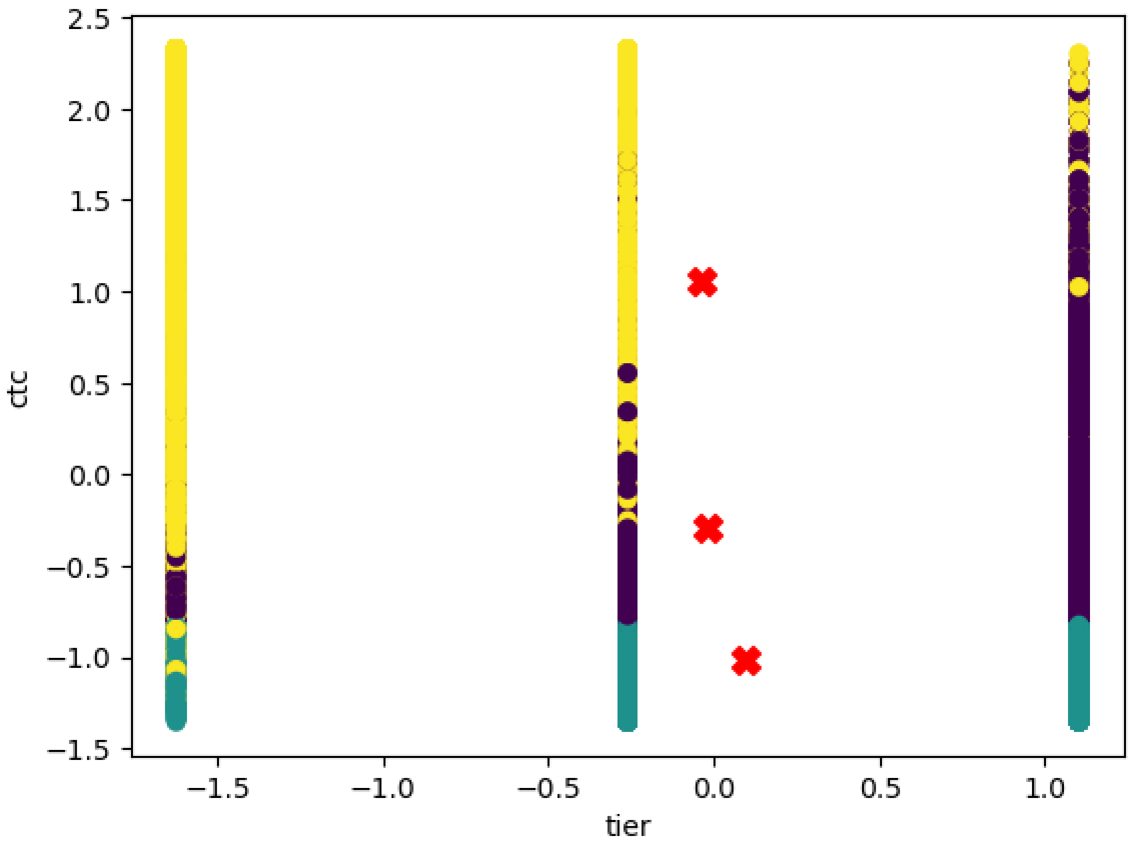
In [104…
```
plt.scatter(clusters['class'], clusters['ctc'], c=clusters['label'])
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], color="re
```

```
plt.xlabel("class")
plt.ylabel("ctc")
plt.show()
```



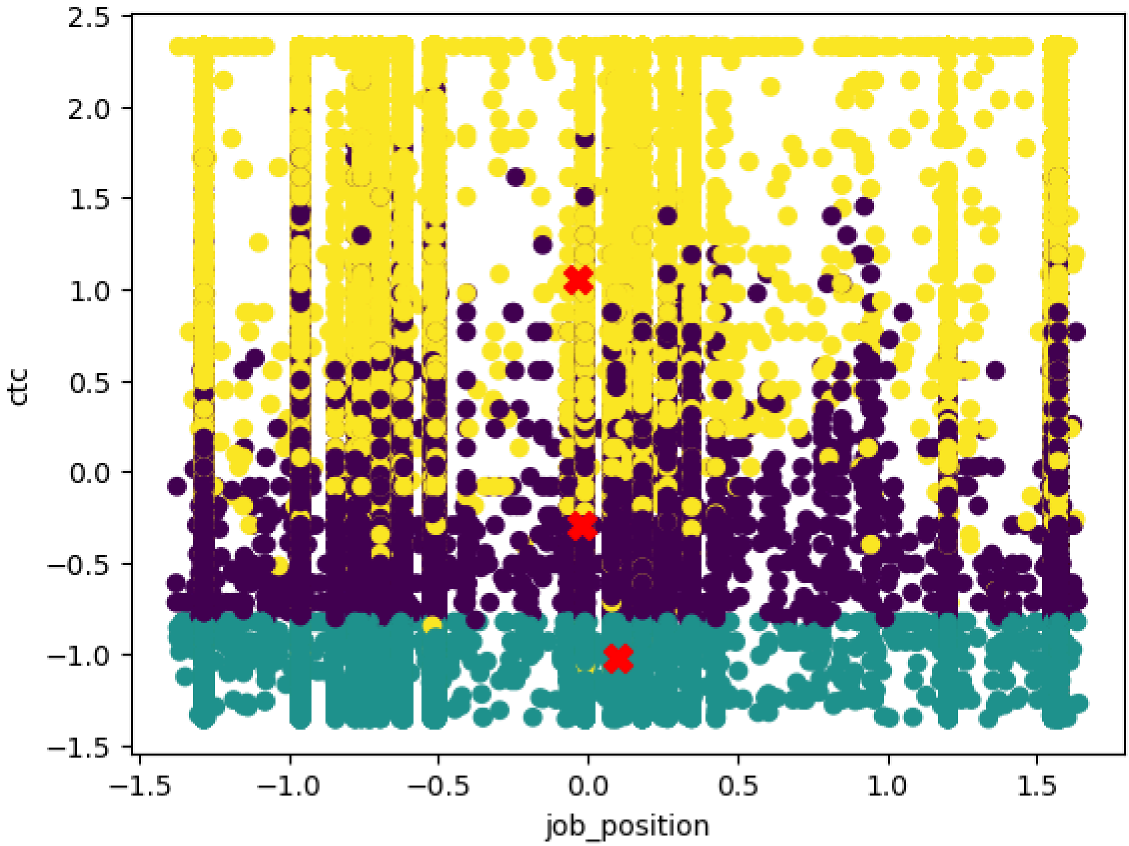
In [105…

```
plt.scatter(clusters['tier'], clusters['ctc'], c=clusters['label'])
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], color="re
plt.xlabel("tier")
plt.ylabel("ctc")
plt.show()
```



In [106…

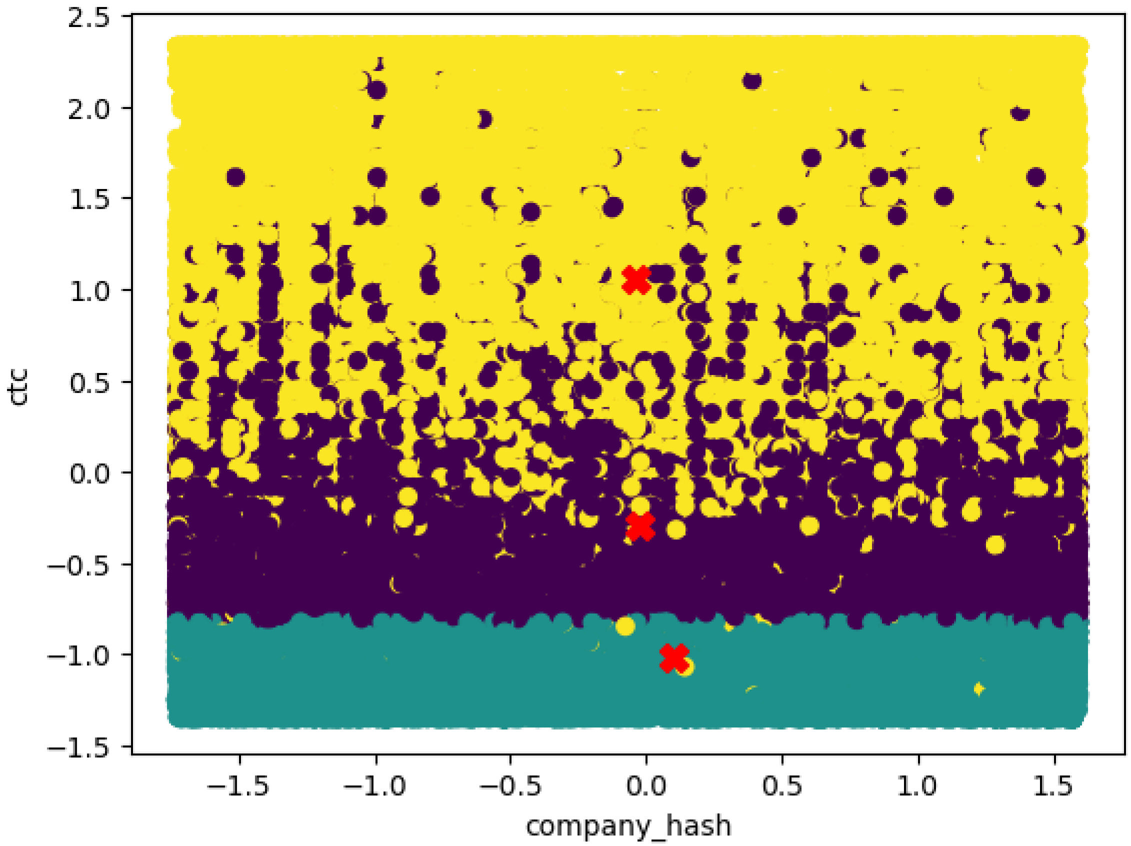
```
plt.scatter(clusters['job_position'], clusters['ctc'], c=clusters['label'])
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], color="re
plt.xlabel("job_position")
plt.ylabel("ctc")
plt.show()
```



In [107…

```
plt.scatter(clusters['company_hash'], clusters['ctc'], c=clusters['label'])
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], color="re
```

```
plt.xlabel("company_hash")
plt.ylabel("ctc")
plt.show()
```



K = 3 is best choice

Hierarchical clustering

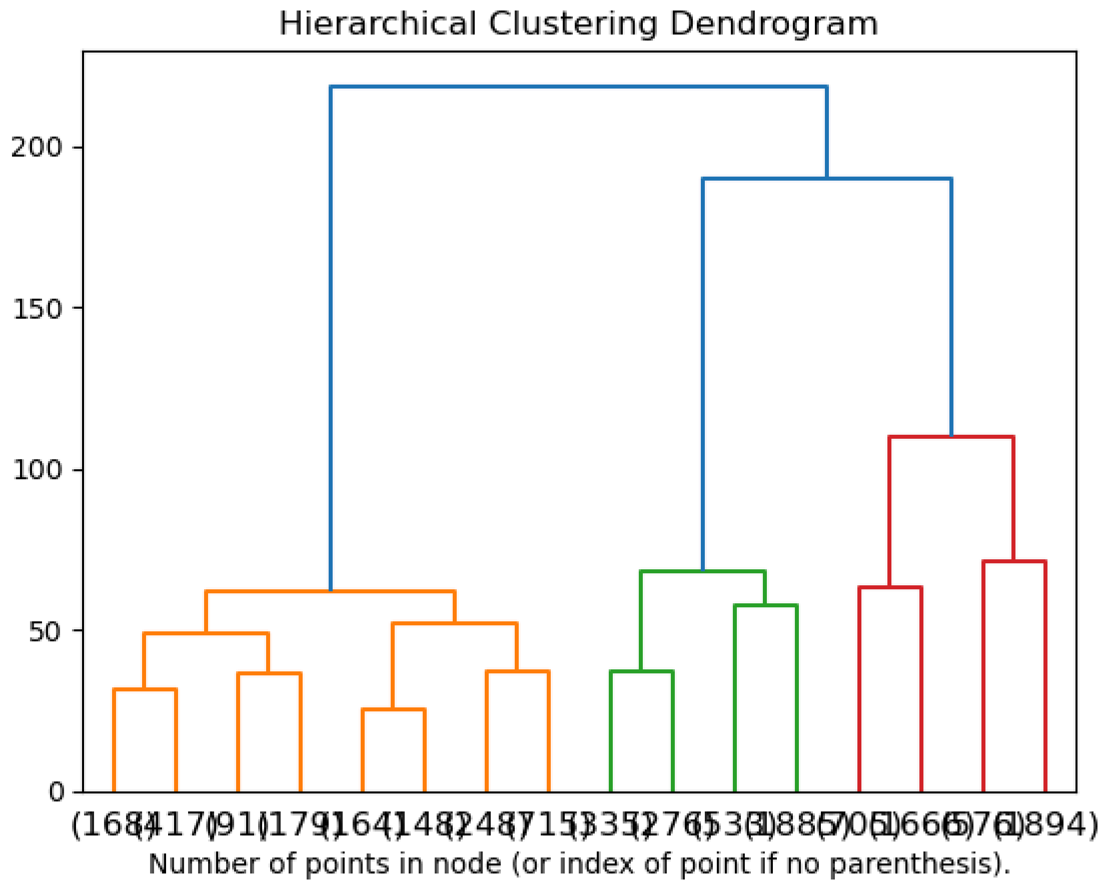
In [108…

```
from sklearn.cluster import AgglomerativeClustering
from scipy.cluster.hierarchy import dendrogram
import scipy.cluster.hierarchy as sch
```

In [109…

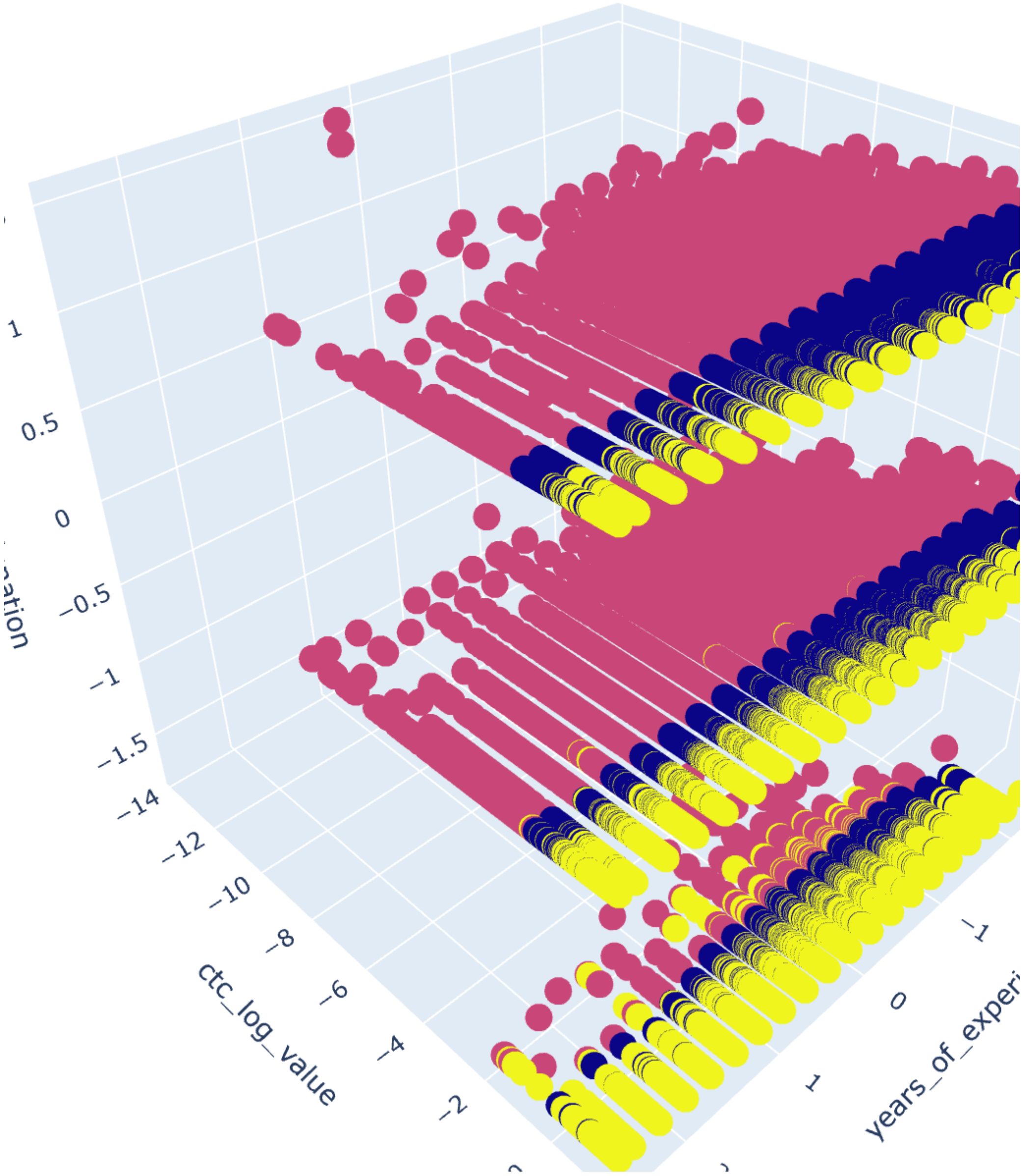
```
# Perform hierarchical clustering
clustering = AgglomerativeClustering(n_clusters=None, distance_threshold=0).fit(X.s

# Plot dendrogram
def plot_dendrogram(model, **kwargs):
    # Create linkage matrix and then plot the dendrogram
    # create the counts of samples under each node
    counts = np.zeros(model.children_.shape[0])
    n_samples = len(model.labels_)
    for i, merge in enumerate(model.children_):
        current_count = 0
        for child_idx in merge:
            if child_idx < n_samples:
                current_count += 1  # leaf node
            else:
                current_count += counts[child_idx - n_samples]
        counts[i] = current_count

    linkage_matrix = np.column_stack([model.children_, model.distances_,
                                      counts]).astype(float)

    # Plot the corresponding dendrogram
```

```python
    dendrogram(linkage_matrix, **kwargs)

plt.title('Hierarchical Clustering Dendrogram')
plot_dendrogram(clustering, truncate_mode='level', p=3)
plt.xlabel("Number of points in node (or index of point if no parenthesis).")
plt.show()
```



In [110…

```python
import plotly.express as px
fig = px.scatter_3d(clusters, x = 'years_of_experience', y = 'ctc_log_value', z = '
                   width = 800, height = 800)
fig.show()
```



Questionnaire

1. What percentage of users fall into the largest cluster?

In [111…

```
clusters["label"].value_counts()/len(clusters)*100

# 45% users belong to cluster 1 which is largest.
```

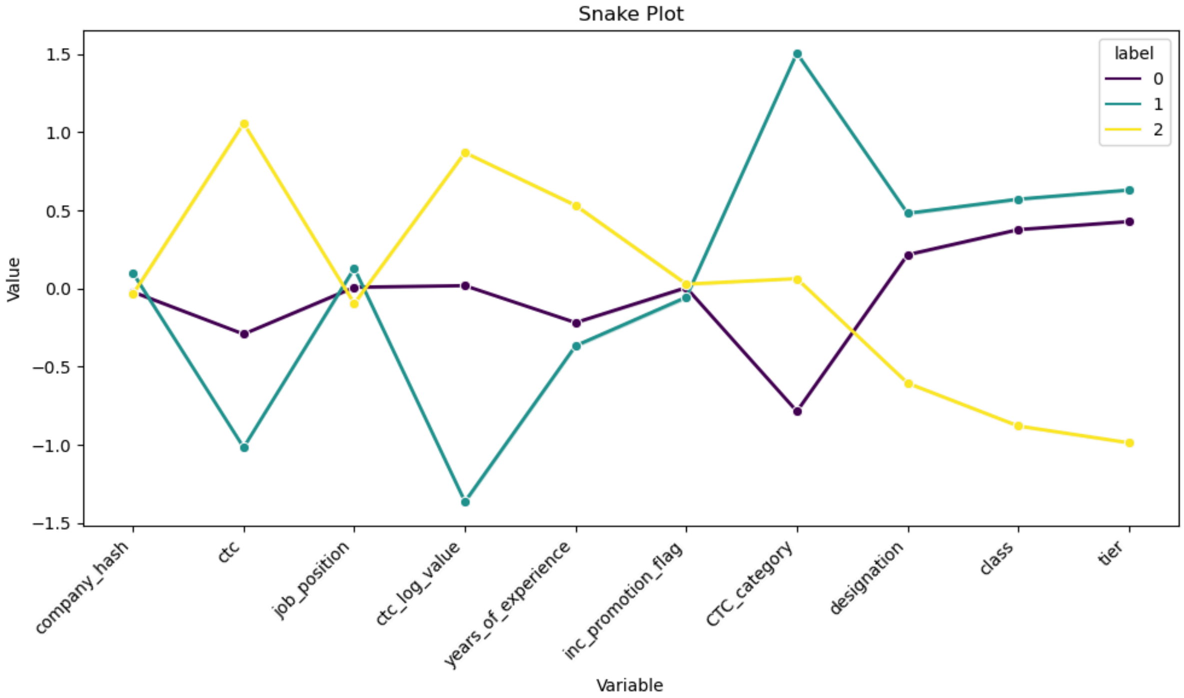
Out[111]:

```
0    44.686038
2    33.402731
1    21.911231
Name: label, dtype: float64
```

2. Comment on the characteristics that differentiate the primary clusters from each other.

In [112…

```
# Melt the DataFrame to long format for visualization
melted_df = pd.melt(clusters, id_vars='label', var_name='years_of_experience')

# Plot the snake plot
plt.figure(figsize=(10, 6))
sns.lineplot(data=melted_df, x='years_of_experience', y='value', hue='label', palet
plt.xticks(rotation=45, ha='right')
plt.title('Snake Plot')
plt.xlabel('Variable')
plt.ylabel('Value')
plt.tight_layout()
plt.show()
```



In [113…

```
# Group by 'label' and 'ctc_category', then count the occurrences
category_counts = clusters.groupby(['label', 'CTC_category']).size()

# Convert the multi-index series to a DataFrame and unstack it to pivot the 'ctc_ca
category_counts_df = category_counts.unstack(level='CTC_category', fill_value=0)
category_counts_df

# Primary cluster is cluster 1, where majority of learners belong to Low ctc catego
# Cluster 0 and 2 shows lot of variation with different features whereas cluster 1
# affected with all features
```

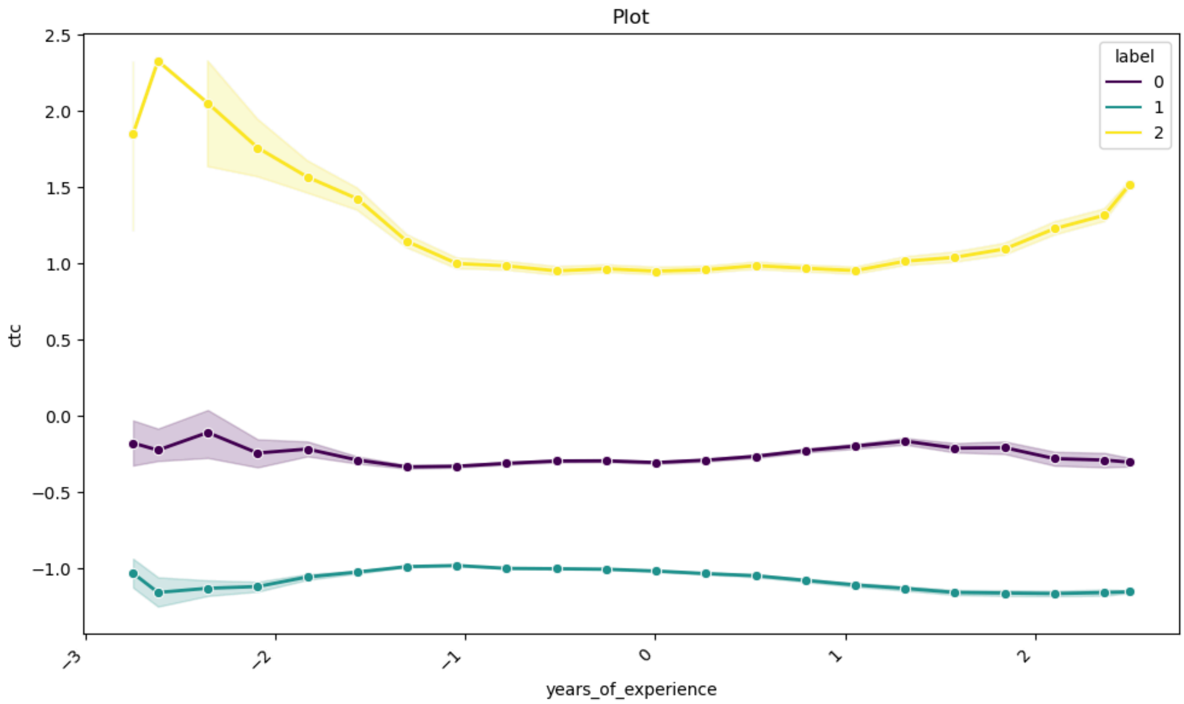
Out[113]:

CTC_category	-1.277139	0.114301	1.505741
label			
0	54418	29719	1
1	0	0	41256
2	2576	60047	270

3. Is it always true that with an increase in years of experience, the CTC increases? Provide a case where this isn't true.

In [114...

```
# Plot the snake plot
plt.figure(figsize=(10, 6))
sns.lineplot(data=clusters, x='years_of_experience', y='ctc', hue='label', palette=
plt.xticks(rotation=45, ha='right')
plt.title('Plot')
plt.xlabel('years_of_experience')
plt.ylabel('ctc')
plt.tight_layout()
plt.show()

# No, It is not always true that with an increase in years of experience, the CTC i
# Observe label 2 cluster, In this cluster, with an increase of experience, the ctc
# infact it decreases for some point of time
```



4. Name a job position that is commonly considered entry-level but has a few learners with unusually high CTCs in the dataset.

In [115...

```
# Define a threshold for "low" years of experience
low_experience_threshold = 1

# Filter the dataset to include only entries with low years of experience
low_experience_df = df[df['years_of_experience'] < low_experience_threshold]

# Group the filtered data by job position and calculate summary statistics for CTC
job_position_stats = low_experience_df.groupby('job_position')['ctc'].describe()

# Sort the positions by mean or median CTC in descending order to find positions wi
high_ctc_positions = job_position_stats.sort_values(by='mean', ascending=False)  #

# Print the job positions with high CTC and their statistics
high_ctc_positions.head(1)

# Job_position 137 has less experience but high ctc.
```

Out[115]:

	count	mean	std	min	25%	50%	75%	max
job_position								
137	2.0	3475000.0	0.0	3475000.0	3475000.0	3475000.0	3475000.0	3475000.0

5. What is the average CTC of learners across different job positions?

In [116…

```
df.groupby("job_position")['ctc'].mean().sort_values(ascending=False)
```

Out[116]:

```
job_position
814    3475000.0
769    3475000.0
357    3475000.0
336    3475000.0
937    3475000.0
          ...
342      10000.0
397      10000.0
291       7500.0
434       2000.0
904       2000.0
Name: ctc, Length: 1006, dtype: float64
```

6. For a given company, how does the average CTC of a Data Scientist compare with other roles?

In [117…

```
data_no_std
```

Out[117]:

	company_hash	orgyear	ctc	job_position	ctc_updated_year	clipped_ctc	clipped_o
0	atrgxnnt xzaxv	2016.0	1100000.0	Other	2020.0	1100000	
1	qtrxvzwt xzegwgbb rxbxnta	2018.0	449999.0	FullStack Engineer	2019.0	449999	
2	ojzwnvwnxw vx	2015.0	2000000.0	Backend Engineer	2020.0	2000000	
3	ngpgutaxv	2017.0	700000.0	Backend Engineer	2019.0	700000	
4	qxen sqghu	2017.0	1400000.0	FullStack Engineer	2019.0	1400000	
...	...	...	...	...	...	...	
188282	vuurt xzw	2008.0	220000.0	nan	2019.0	220000	
188283	husqvawgb	2017.0	500000.0	nan	2020.0	500000	
188284	vwwgrxnt	2021.0	700000.0	nan	2021.0	700000	
188285	zgn vuurxwvmrt	2019.0	3475000.0	nan	2019.0	5100000	
188286	bgqsvz onvzrtj	2014.0	1240000.0	nan	2016.0	1240000	

188287 rows × 14 columns

In [118…

```
# Filter the dataset to include only entries for the given company
company_df = data_no_std[data_no_std['job_position'] == "Data Scientist"]

# Group the filtered data by job position and calculate the average CTC for each gr
job_position_ctc_avg = company_df.groupby('job_position')['ctc'].mean().reset_index

# Print the average CTC for each job position
print(job_position_ctc_avg)

# Compare the average CTC of the Data Scientist role with other roles
data_scientist_avg_ctc = job_position_ctc_avg[job_position_ctc_avg['job_position']
other_roles_avg_ctc = job_position_ctc_avg[job_position_ctc_avg['job_position'] !=

# Print the comparison
print(f"Average CTC of Data Scientist: {data_scientist_avg_ctc}")
print(f"Average CTC of Other Roles: {other_roles_avg_ctc}")
```

```
     job_position          ctc
0  Data Scientist  1.395708e+06
Average CTC of Data Scientist: 1395708.0671810312
Average CTC of Other Roles: nan
```

7. Discuss the distribution of learners based on the Tier flag, Which companies dominate in Tier 1

In [119…

```
# Step 1: Calculate the mean CTC for Tier 1
mean_ctc_tier1 = df[df['tier'] == 1]['ctc'].mean()

# Step 2: Filter the dataset to include only entries for Tier 1
tier1_df = df[df['tier'] == 1]

# Step 3: Compare the CTC of each company in Tier 1 with the mean CTC
companies_higher_ctc = tier1_df[tier1_df['ctc'] > mean_ctc_tier1]['company_hash'].u

companies_higher_ctc

# Companies with these codes dominate in tier1.
```

Out[119]:

```
array([13830, 20209, 21008, ...,  9137, 17867, 32028])
```

8. After performing unsupervised clustering:

1. How many clusters have been identified using the Elbow method?

2. Do the clusters formed align or differ significantly from the manual clustering efforts? If so, in what way?

1. Using Elbow method, 3 clusters have been identified.
2. clusters formed differ significantly from manual clustering efforts because a. as data is too large, manual clustering takes too much time. b. manual clustering has limited perpective. c. we didn't have elbow method by which we can easily determine the best choice of clusters. d. manual clustering may lack showing hidden patterns.

Insights And Recommendation

Insights

1. The dataset consists of 205,843 entries with 7 features, including company information, CTC, job position, and update year.
2. There were missing values in the dataset which was handled by imputation mean, and features have different data types, including integers, floats, and objects.
3. The majority (47.7%) of learners earn a high CTC, followed by average (30.3%) and low (22.1%) CTCs.
4. A significant portion (56.8%) of learners received both promotion and increment in CTC, while others experienced changes in job roles or received increments only.
5. Using the Elbow method, 3 clusters have been identified. Cluster 1 is the largest, primarily consisting of learners with low CTC.
6. There's no direct correlation between years of experience and CTC, especially evident in Cluster 2, where CTC decreases with increasing experience for some time.
7. Manual clustering efforts were challenging due to the large dataset size, limited perspective, and lack of clear patterns. However, clustering based on designations, classes, and tiers revealed insights into CTC distributions.
8. There are 856 unique job positions and 37,180 unique company hashes in the dataset, with backend engineer being the most common job position.
9. Numerical columns were clipped and log-transformed to handle outliers and skewed distributions effectively.
10. There's a strong correlation between tier and CTC, with learners in Tier 3 consistently having low CTC regardless of experience.

Recommendation

1. Further analysis can be conducted to understand the factors influencing CTC distribution, such as industry trends, job demand, and geographical location.
2. Regularly monitor and update clustering models to adapt to changing trends and patterns in the data.
3. Seek insights from domain experts to enhance understanding of the dataset and identify relevant features for analysis.
4. Advanced visualization techniques can be done to gain deeper insights into the data and effectively communicate findings to stakeholders
5. Robustness and reliability of results can be ensured by further evaluating clustering models using different algorithms and validation techniques.
6. Predictive modeling techniques can be expolored to forecast CTC trends and identify factors contributing to variations in CTC distribution.
7. Foster a culture of continuous improvement in data analysis practices, incorporating feedback and lessons learned from previous analyses.

In [ ]:

In [ ]:

In [ ]:

In [ ]:

```
In [ ]:
```